
Universidade de Aveiro
2010
Departamento de Electrónica, Telecomunicações e Informática

Tiago Manuel Henriques Carrão

Plataforma de medida e caracterização de redes sem fios


Universidade de Aveiro
2010
Departamento de Electrónica, Telecomunicações e Informática

Tiago Manuel Henriques Carrão

Plataforma de medida e caracterização de redes sem fios

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Diogo Gomes, Assistente Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor Rui Aguiar, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho à minha família e aos meus amigos.

O júri

Presidente	**Prof. Dr. João Nuno Matos** Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Vogal	**Prof. Dr. Diogo Gomes** Assistente Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (Orientador)
Vogal	**Prof. Dr. Rui L. Aguiar** Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro (Co-orientador)
Vogal	**Prof. Dr. Paulo Carvalho** Professor Auxiliar do Departamento de Informática da Universidade do Minho

Agradecimentos

Aproveito este espaço para, em breves linhas, revelar o meu apreço e gratidão a todas as pessoas que, directa ou indirectamente, tiveram influência no resultado final deste trabalho.

Ao meu orientador, Professor Diogo Gomes, pela possibilidade de realizar este trabalho. Ao Professor João Paulo Barraca, pela orientação e acompanhamento ao longo do desenvolvimento deste projecto. A Luca Deri, criador da *nProbe*, pela disponibilidade demonstrada. Ao grupo de pessoas que compõe o HNG pela sua colaboração.

Aos meus amigos e todas as outras pessoas que contribuíram não só para este trabalho, mas também para a minha formação enquanto aluno e pessoa, de entre as quais destaco Fábio Ferreira, Carlos Pereira e Joana Silva.

Aos meus pais e irmão por dia após dia continuarem a transmitir todo o apoio e carinho, ajudando-me ao longo do meu percurso académico.

Palavras-chave Redes sem fios, Métricas, Redes de Teste, OMF, OML, AMazING

Resumo

Tem-se assistido nos últimos anos a um crescente interesse nas tecnologias de redes sem fios que, por consequência, têm sido alvo de um estudo e desenvolvimento cada vez maiores. Surge assim a necessidade de testar as várias tecnologias e configurações desenvolvidas com uma cada vez maior transparência, rigor e replicabilidade.

Neste contexto existem três soluções para o estudo de tecnologias de rede, baseadas em simulações, emulações e redes de teste. É nesta última possibilidade que irá incidir esta dissertação, mais nomeadamente na perspectiva de recolha de métricas de rede e caracterização da rede sem fios.

Neste trabalho é analisado o estado da arte actual, abordando os processos utilizados e as propostas relevantes para esta área, com particular destaque para a biblioteca de medidas OML.

É proposta a implementação de uma extensão IPFIX à biblioteca OML, permitindo a recolha de métricas através deste protocolo de transporte, sendo posteriormente armazenadas numa base de dados.

A solução apresentada é exposta em detalhe e analisada ao longo de testes, com foco nas métricas taxa de transferência, perda de pacotes e *jitter*.

Por fim, a implementação desenvolvida é comparada com a biblioteca OML, a nível de resultados obtidos e de consumo de recursos do sistema.

Keywords Wireless Networks, Metrics, *Testbeds*, OMF, OML, AMazING

Abstract

In past recent years there has been a rising interest in wireless networks technology, consequently becoming the target of an increasing development and study. Therefore, there is a greater demand for the ability to test the various developed technologies and configurations, with transparency, accuracy and replicability.

In this context there are three solutions to study networks technology, based on simulations, emulations and testbeds. This dissertation is about the latter, specifically on the gathering of network metrics and characterization of the wireless network.

In this work it is done an analysis of the current state of the art, approaching the used methods and relevant proposals in this area, with particular emphasis on the measurement library OML.

It is presented the implementation of an IPFIX extension to the OML library, enabling the collection of network metrics through this transport protocol, for later storage in a database.

The presented solution is exposed in detail and analysed through tests, focusing on the metrics throughput, packet loss and jitter.

Finally, the developed implementation is compared to the OML library, regarding the obtained results and the system's resources consumption.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

AD – Application Definiton

AM – Aggregate Manager

AMazING – Advanced Mobile wIreless Network playground

BGP – Border Gateway Protocol

CPU – Central Processing Unit

D-ITG – Distributed Internet Traffic Generator

EC – Experiment Controller

ED – Experiment Description

ETT – Expected Transmission Time

ETX – Expected Transmission Count

EUA – Estados Unidos da América

GPS – Global Positioning System

HTTP – Hypertext Transfer Protocol

IAT – Inter Arrival Time

ICMP – Internet Control Message Protocol

ID - Identification

IDT – Inter Departure Time

IEEE – Institute of Electrical and Electronics Engineers

IETF – Internet Engineering Task Force

IP – Internet Protocol

IPFIX – Internet Protocol Flow Information eXport

IT – Instituto de Telecomunicações

ITU-T – Telecommunication Standardization Sector

MAP – Mesh at Purdue

MGEN – Multi-Generator

MIT – Massachusetts Institute of Technology

ML – Measurement Library

MPLS – Multiprotocol Label Switching

NFS – Network File System

NIC – Network Interface Card

NS-2 – Network Simulator

OEDL – OMF Experiment Description Language

OMF – cOntrol and Management Framework

OML – OMF Measurement Library

PC – Personal Computer

PD – Prototype Definition

PLR – Packet Loss Rate

PoE – Power over Ethernet

PS – Packet Size

RAM – Random-Access Memory

RC – Resource Controller

RPC – Remote Procedure Calls

RTP – Real-Time Transport Protocol

RTT – Round Trip Time

SIP – Session Initiation Protocol

SQL – Structured Query Language

TCP – Transmission Control Protocol

ToS – Type of Service

TTL – Time To Live

UCR – University of California, Riverside

UCSB – University of California, Santa Barbara

UDP – User Datagram Protocol

UMIC – Ultra high-speed Mobile Information and Communication Systems

VLC – VideoLan Client

WCETT – Weighted Cumulative Expected Transmission Time

XML – Extensible Markup Language

Capítulo 1 - Introdução

1.1 Motivação

As redes sem fios têm sofrido uma das maiores evoluções no panorama tecnológico dos últimos anos, dada a sua enorme gama de aplicações, sendo as redes sem fio da família IEEE 802.11 [1] o padrão para as redes sem fios locais. É portanto de grande interesse a avaliação do desempenho de algoritmos e protocolos. Simulação, emulação e experimentação são três das técnicas mais comuns no domínio da análise de desempenho em redes sem fios [2].

Simulação é a abordagem mais popular, eficaz e viável para testar protocolos em redes de testes. As ferramentas de simulação (e.g. NS [3]) tentam recriar e prever o comportamento de ambientes reais em diferentes cenários. Apesar de apresentarem vantagens como um custo mais baixo, facilidade e flexibilidade de desenvolvimento, as ferramentas de simulação pecam por utilizarem modelos simplificados, que não conseguem traduzir fielmente as condições de um ambiente real, fornecendo resultados distantes da realidade, válidos para uma primeira aproximação, mas insuficientes para uma análise mais aprofundada.

Emulação é uma solução híbrida que combina *hardware* e *software* com o objectivo de imitar o comportamento de uma rede com alguns dos seus componentes implementados em ambiente real e outros simulados. Esta alternativa oferece assim replicabilidade, um controlo apertado sobre a experiência e um certo grau de realismo. No entanto, também esta solução fica aquém das complexidades do mundo real falhando por vezes a tentativa de recriar os efeitos causados por um ambiente real numa rede sem fios [2].

Redes de experimentais [4] são a resposta às incapacidades dos dois métodos anteriores, que apesar de serem ferramentas pertinentes na avaliação de desempenho de redes levam por vezes a resultados imprecisos devido à utilização de pressupostos irrealistas ou modelos demasiado simplificados. As redes de teste tentam assim fazer a ponte entre simulação e o mundo e a implementação real (ver Figura 1 [5]), proporcionando um ambiente para testes semelhante ao real, com a possibilidade de configuração, monitorização e execução de experiências remotamente. Contudo levantam-se algumas questões em relação às redes de teste, tais como custos mais elevados, quer de desenvolvimento quer de manutenção, dificuldades na reutilização da estrutura desenvolvida, uma vez que grande parte das redes de teste são criadas para uma experiência em particular, perdendo a utilidade finda a experiência e não podendo ser utilizada para novas experiências com diferentes especificidades.

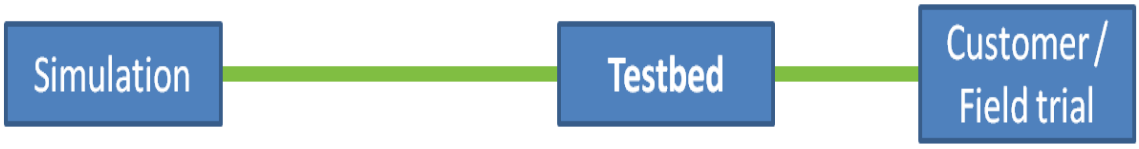

Figura 1 - Ponte de ligação entre simulação e ambiente real

É então necessário o desenvolvimento de plataformas de teste mais genéricas que permitam implementar protótipos em ambientes reais controlados, que permitam correr várias experiências, com resultados bastante próximos da realidade. A plataforma de gestão de redes de testes OMF [6], recorrendo à biblioteca de medidas OML [7], vem colmatar algumas destas falhas, proporcionando um conjunto de ferramentas para instrumentação e execução de uma experiência, recolha de resultados e gestão dos recursos da rede.

Todavia a biblioteca de medidas OML possui algumas restrições, de entre as quais se destacam a necessidade de haver da parte do utilizador o tratamento das aplicações a testar de forma a acrescentar pontos de medida, de onde a plataforma recolherá os valores pretendidos e o conjunto limitado de métricas possíveis de extrair.

É neste contexto que esta dissertação será desenvolvida, simplificando o processo de recolha de métricas numa rede de testes – eliminando a necessidade de alterar as aplicações a testar - e alargando o leque de métricas possíveis de avaliar, com a introdução do protocolo de transporte IPFIX [8] na biblioteca OML.

1.2 Objectivos

Esta dissertação tem como principal objectivo o desenvolvimento e implementação de uma arquitectura de recolha, armazenamento e visualização de resultados numa rede experimental, da qual outros utilizadores se poderão servir. Procura-se criar uma alternativa à actual plataforma de medidas OML, recorrendo a um protocolo de exportação de fluxos, IPFIX, para a aquisição de métricas de desempenho da rede, que permitam efectuar uma caracterização da mesma.

Inerentemente a este objectivo principal da dissertação existem outros objectivos secundários, como a investigação de redes de testes com características semelhantes, a determinação de métricas de operação e rede a obter e um estudo de ferramentas para a realização de testes de desempenho na rede e medidas.

1.3 Estrutura da Tese

Esta dissertação está dividida em capítulos, com a seguinte organização:

No Capítulo 2 é feito um reconhecimento das redes de testes actualmente existentes e identificadas as mais interessantes numa óptica de captura de métricas.

No Capítulo 3 é efectuado um apanhado das métricas relacionadas com redes sem fios, bem como as estratégias para sua captura, tanto locais como remotas.

No Capítulo 4 é realizada uma abordagem às ferramentas disponíveis para realização de medidas e testes de desempenho da rede, sendo apresentada a plataforma base deste trabalho, a OML.

No Capítulo 5 é apresentada a solução proposta, dividida em duas secções, colector e exportador. É ainda descrita a sua implementação e integração.

No Capítulo 6 são identificadas as métricas obtidas, analisados os resultados capturados e é feita uma comparação entre os resultados obtidos através da solução actual e através da solução proposta.

Por fim, no Capítulo 7, é feita a conclusão em que se avalia o trabalho realizado.

Capítulo 2 – Redes de Testes

Uma rede de testes é uma plataforma para a experimentação de novas aplicações e protocolos de rede desenvolvidos. As redes experimentais permitem realizar experiências de uma forma rigorosa, transparente e replicável. Como referido na secção 1.1 um ambiente de testes virtual proporciona aos investigadores a possibilidade de testar novas tecnologias num ambiente controlado, configurável e bastante próximo do real [9].

As redes de testes servem assim como veículo de ligação entre a simulação, cujas simplificações dos modelos matemáticos utilizados ficam aquém das complexidades das redes sem fios reais, e o ambiente real, que não concede os níveis de controlabilidade e replicabilidade necessários para a experimentação de novas tecnologias de redes.

De momento, existem a nível global várias redes de teste nas mais diversas localizações, como se pode atestar na Figura 2.

Figura 2 - Mapa do Mundo com a localização de várias redes experimentais

Estas redes de teste assentam em plataformas criadas com diferentes propósitos e portanto com distintas particularidades, quer em relação às características dos nós, quer em relação às métricas registadas e ao seu processo de medição e recolha.

Do universo de redes experimentais existentes, merecem especial destaque a UMIC Mesh [10], que possui uma ferramenta própria para extracção de métricas, Roofnet [11] do MIT [12], que levou à introdução de uma nova métrica complexa (E*xpected Transmission Count* [13]), Orbit [14], rede de testes que deu origem à plataforma OMF e à biblioteca OML, sendo esta última a base do trabalho desenvolvido e, por fim, AMazING [15], a rede onde será implementada a solução criada.

Para além das redes experimentais previamente mencionadas existem muitas outras, que não serão alvo de uma análise mais detalhada como as anteriores, por várias razões como o facto de possuírem características e objectivos diferentes dos deste projecto e/ou falta de informação, das quais são exemplos MAP - Purde [16], desenvolvida para facilitar quer o estudo de redes sem fios quer para experimentar conceitos teóricos e a sua aplicabilidade; BerlinRoofNet [17], uma rede de testes implementada em Berlim baseada na Roofnet do MIT, com o objectivo similar de fornecer uma rede comunitária em toda a cidade; Moment [18], rede criada para facilitar a experimentação de protocolos no laboratório de redes da Universidade da Califórnia – Santa Bárbara [19]; BWN-Mesh [20], é uma rede de apoio ao laboratório de redes do Instituto de Tecnologia da Geórgia [21] (EUA) no estudo protocolos de rede adaptativos; UCR [22], rede da Universidade da Califórnia – Riverside [23], que reclama para si o título de primeira rede de testes a adoptar uma arquitectura combinada de Linux NFS [24] e PoE [25] com plataformas de *hardware* sem fios.

2.1 UMIC-Mesh.net

UMIC-Mesh.net é uma rede de testes híbrida sem fios, da autoria do grupo de investigação UMIC [26], que consiste em 51 nós reais e um ambiente de teste virtual.

Esta abordagem permite por um lado o desenvolvimento e teste de *software* num ambiente virtual, mas de uma forma controlável e replicável; por outro lado, os resultados e as conclusões tirados podem ser facilmente transpostos para a realidade, dado o elevado grau de realismo que a rede de testes oferece.

Na óptica da recolha de estatísticas de rede, esta rede experimental distingue-se pelo uso da aplicação *Flowgrind* [27], que se trata de uma ferramenta semelhante ao Iperf [28] e Netperf [29] com a particularidade de possuir uma arquitectura distribuída (ver Figura 3 [30]), capaz de medir várias métricas: taxa de transferência, *round-trip time*, perda de pacotes, entre outras.

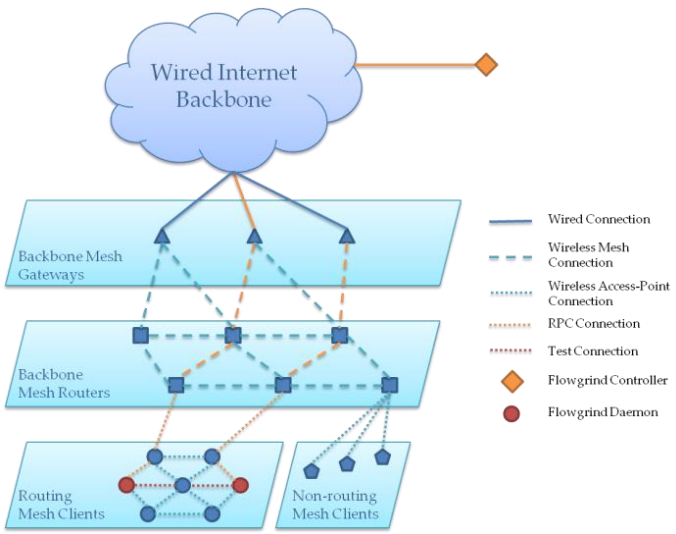

Figura 3 - Arquitectura do Flowgrind

A arquitectura da ferramenta *Flowgrind*, Figura 3, é dividida em duas partes: o controlador, responsável pela configuração dos testes e disponibilização de resultados, e pelo *daemon*, que realiza os testes. Como se pode verificar na Figura 3, os testes são configurados nos controladores, que não participam nos testes activamente. Os *Flowgrind daemons* executam os testes e comunicam os resultados para o controlador, através de uma ligação RPC [31], que são assim agregados na localização do controlador, que não necessita de fazer parte da rede de testes.

2.2 Roofnet – MIT

Este projecto é constituído por uma rede de 54 nós, distribuídos ao longo de 2 km (ver Figura 4 [11]). Cada nó é baseado num PC a correr Linux, uma NIC IEEE 802.11b e uma antena omnidireccional.

Um dos objectivos principiais desta rede é fornecer acesso à rede do campus e à Internet duma forma barata, através da comunicação por saltos entre os vários nós até aos *gateways* da Universidade.

Esta rede experimental oferece a possibilidade de extrair métricas como a taxa de transferência, taxa de perda de pacotes, RTT, número de saltos e ainda ETX, uma métrica complexa traçada no MIT, alvo de estudo nesta dissertação na secção 3.2.1.

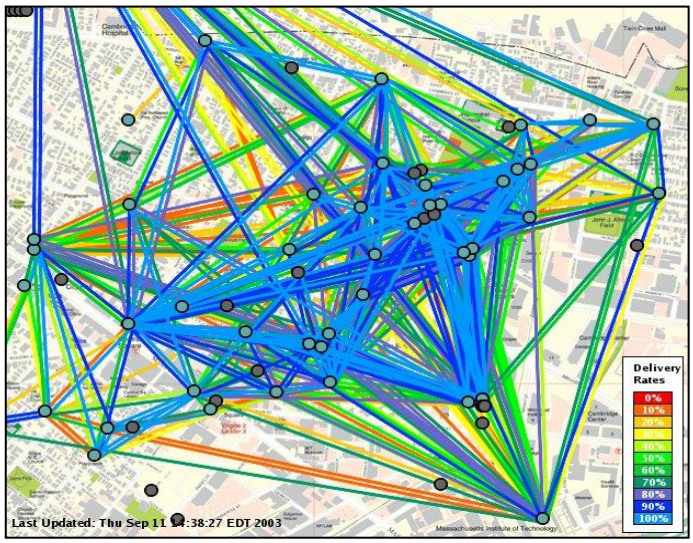

Figura 4 - Disposição dos nós da rede RoofNet

2.3 Orbit

A rede experimental Orbit, iniciativa do Winlab [32] é composta por 400 nós, dispostos numa malha de duas dimensões.

Esta rede foi criada com o objectivo de proporcionar reprodutibilidade experimental aliada ao suporte à avaliação de protocolos e aplicações em ambientes reais.

É este projecto que dá origem à plataforma de gestão e controlo OMF e consequentemente à biblioteca de medidas OML, que serão posteriormente descritas nas secções 4.1 e 4.2, respectivamente, que foram depois melhoradas tornando possível a sua integração noutras redes de testes. Estas ferramentas surgiram numa tentativa de dar resposta à falta de ferramentas e metodologias que forneçam descrições completas de experiências, incluindo recursos utilizados e medições a efectuar, com o objectivo de aumentar o rigor científico na área das redes.

A rede Orbit permite, com recurso às plataformas OMF e OML, a captura de várias métricas de rede: taxa de transferência, taxa de perda de pacotes e *jitter* [33].

A Figura 5 [34] demonstra o processo de experimentação na rede Orbit. O utilizador, à direita, fornece a descrição da experiência e os resultados a recolher. Essa descrição permite a configuração automática dos recursos da rede necessários e os valores a medir. No final da experiência os resultados obtidos são armazenados e disponibilizados ao utilizador.

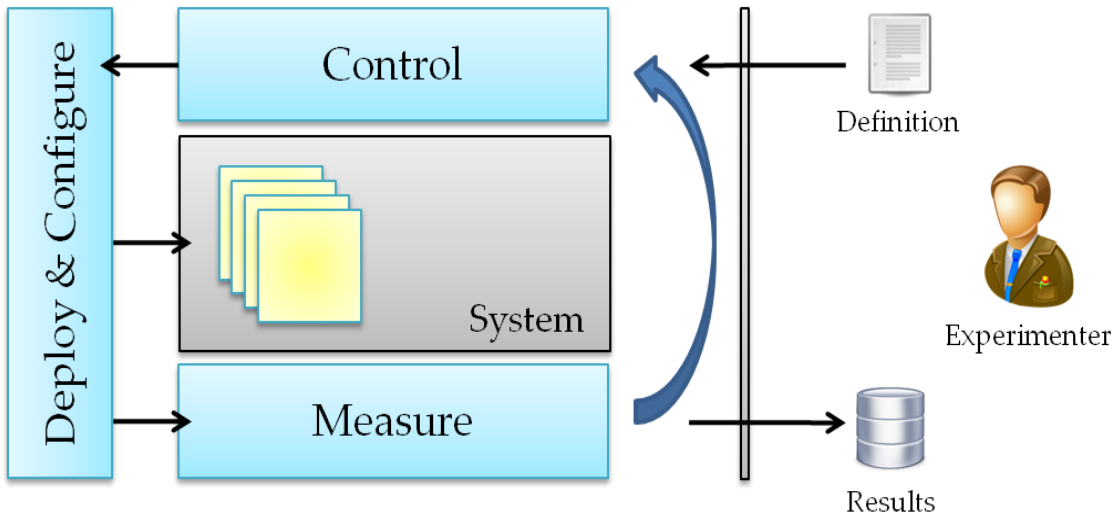

Figura 5 - Modelo de experimentação na Orbit

2.4 AMazING

É nesta rede de testes que irá incidir esta dissertação, sendo o objectivo a sua caracterização através da solução desenvolvida ao longo deste trabalho. A AMazING é formada por 24 nós colocados no terraço do Instituto de Telecomunicações (IT) de Aveiro [35], de acordo com a disposição na Figura 6 [36], num ambiente de reduzida interferência electromagnética.

Posteriormente será adicionado o suporte à mobilidade, com a inclusão de um nó móvel ao longo de um *monorail*, atingindo velocidades de 60 km/h, com dois módulos: o primeiro, semelhante aos componentes dos outros nós; e o segundo, uma plataforma que confere a mobilidade necessária, incorporando um motor DC, GPS e hodómetro.

Esta rede experimental recorre à plataforma OMF e a biblioteca OML para as tarefas de gestão, controlo da rede e recolha de medidas, tendo portanto a capacidade de obtenção de métricas semelhantes à rede Orbit: taxa de transferência, taxa de perda de pacotes e *jitter*.

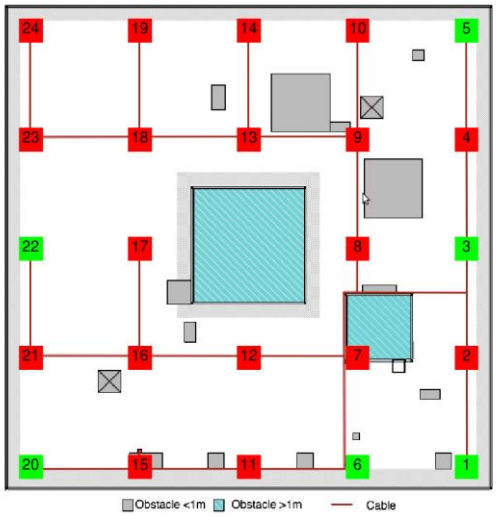

Figura 6 - Rede experimental AMazING

2.5 Comparação das Redes Experimentais

Na Tabela 1 encontram-se sumariadas as redes experimentais analisadas, numa perspectiva de métricas passíveis de extracção e o processo para tal.

Métricas / Redes	Taxa de Transferência	Atraso	Taxa de Perda de Pacotes	*Jitter*	Outras	Método
UMIC	✓	✓	✓	✗	IAT, congestionamento TCP	*Flowgrind*
MAP - Purdue	✓	?	?	?	Congestionamento TCP	?
Roofnet	✓	✓	✓	✗	ETX, Número de saltos	?
Orbit	✓	✗	✓	✓	✗	OML
AMazING	✓	✗	✓	✓	Métricas tiradas pela *nProbe* [37] (ver Apêndice F)	OML modificada
BerlinRoofNet	✓	✓	?	?	?	?
Moment	✓	✓	✗	✓	*Overhead*, congestionamento	DAMON [38]
BWN-Mesh	✓	✓	?	?	?	?
UCR	✓	✗	✓	✓	✗	*Iperf*

Tabela 1 - Comparativo entre as Redes de Testes

Ao analisar a Tabela 1 é possível estabelecer algumas comparações sobre as capacidades das redes experimentais estudadas.

Ao comparar as métricas possíveis de obter em cada rede de testes constata-se que um factor comum entre todas as redes é a possibilidade de medir a taxa de transferência; *jitter* é a métrica com menos possibilidades de obter, apenas possível em quatro redes; atraso e a taxa de perda de pacotes são facultados por cinco redes de testes cada.

Algumas das redes de testes oferecem ainda a possibilidade de outros resultados, como ETX, número de saltos e congestionamento.

Na perspectiva das redes experimentais, conclui-se que cada uma fornece uma gama diferente de métricas, sendo as redes mais completas a rede UMIC e Roofnet (onde é possível obter

taxa de transferência, atraso e perda de pacotes), Orbit, AMazING e UCR (taxa de transferência, perda de pacotes, *jitter*) e Moment (taxa de transferência, atraso e *jitter*).

Por fim, constata-se que cada rede possui um processo de recolha de métricas distinto. UMIC possui a ferramenta *Flowgrind* enquanto Orbit e AMazING a OML, por exemplo.

Capítulo 3 -Métricas

Este capítulo apresenta um conjunto de métricas utilizadas para caracterizar e descrever o comportamento de uma rede em relação à sua utilização, desempenho e fiabilidade.

As métricas permitem avaliar a performance de uma rede ao nível da camada 3 (camada IP) e devem ser independentes da metodologia usada para as medir, sendo expressas em unidades padrão, como por exemplo bps - bits por segundo [39].

Nas secções seguintes estão definidas as várias métricas estudadas, bem como a metodologia utilizada para as adquirir, divididas nos seguintes grupos:

- **Métricas Elementares**

  Este grupo de métricas corresponde às métricas cuja determinação é mais simples, i.e. são possíveis de adquirir de uma forma instantânea, e sem necessidade de cálculos elaborados [40]. Tendo isso em conta, num cenário de uma rede dinâmica, estas métricas podem-se revelar mais pertinentes que as complexas. Estas métricas podem ainda ser utilizadas na estimativa de métricas mais complexas. São exemplos, o número de saltos, taxa de transferência e o número de pacotes perdidos.

- **Métricas Complexas**

  Contrariamente ao grupo de métricas anteriores estas requerem um tempo de amostragem superior e vários parâmetros de rede para serem calculadas. Conseguem assim alcançar um maior nível de precisão, permitindo uma análise da rede mais aprofundada. Contudo podem ser preteridas em prol de métricas elementares em cenários de maior mobilidade. Fazem parte deste grupo métricas como ETX e WCETT [41].

3.1 Métricas Elementares

3.1.1 Número de Saltos

Trata-se de uma das mais básicas métricas possíveis de adquirir e resume-se a uma simples medida do número de saltos entre a origem e o destino de uma rota, sendo considerado um salto cada *router* atravessado ao longo da rota [40].

No entanto, o número de saltos oferece uma visão bastante limitada da qualidade da ligação de rede ao ignorar aspectos como a carga na rede e a qualidade e capacidade da ligação, diversidade de canais e outras especificidades dos nós.

Em [40] foi demonstrado que uma rota com um maior número de ligações mais curtas oferece melhorias significativas de desempenho sobre uma rota composta por um menor número de ligações de maior distância e consequentemente menor qualidade.

Ainda de acordo com [42] esta métrica tem a tendência a escolher rotas através de alguns nós com localização centralizada, gerando congestionamento na rede.

Apesar de nenhuma das redes experimentais oferecer a possibilidade integrada de obtenção do número de saltos esta é uma métrica que pode ser adquirida através de um simples comando *traceroute*, que consiste no envio de pacotes UDP [43] com um TTL inicial de 1 de maneira a que ao atingirem o primeiro nó no caminho para o destino sejam retornadas respostas ICMP [44] de tempo excedido, que transportam o endereço dos nós correspondente ao primeiro salto. O TTL dos próximos pacotes é incrementado de 1, alcançando o segundo nó do caminho pretendido, que responde com ICMP de tempo excedido, e assim sucessivamente até ser suficiente para alcançar o nó correspondente ao endereço de destino, que retorna uma resposta ICMP de porta inalcançável do endereço de destino. Para garantir o não processamento dos pacotes UDP pelo nó de destino são utilizadas portas de destino com valores improváveis. A Figura 7 é exemplo deste processo.

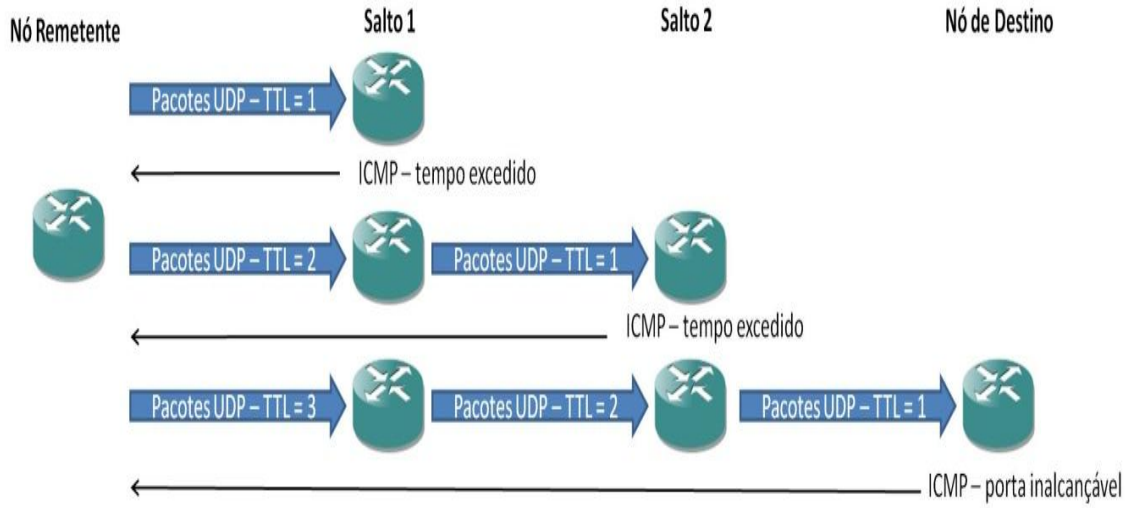

Figura 7 - Exemplo do cálculo do número de saltos

Em suma, embora não seja uma métrica optimizada para estabelecer rotas com altas taxas de transferência [13], o número de saltos pode-se revelar bastante útil e superior a métricas dependentes da carga na rede em cenários dinâmicos [45].

3.1.2 Taxa de Transferência

Também designada por largura de banda alcançável, esta métrica é a quantidade máxima de dados por unidade de tempo transmitida com sucesso entre dois nós através de uma ligação ou uma rota, sendo influenciada pela utilização actual, o protocolo e sistema operativo utilizados e o desempenho do nó destino em termos de capacidade e carga [46]. A taxa de transferência é usualmente expressa em bits por segundo.

A taxa de transferência pode assumir diferentes valores, consoante o contexto pretendido: taxa de transferência teórica, que são valores teóricos ou calculados da quantidade máxima que pode ser transmitida em condições ideais; taxa de transferência medida, é a taxa de transferência medida num sistema real ou simulado, medida ao longo de um curto período de tempo; taxa de transferência sustentada, que corresponde ao valor da média da taxa de transferência ao longo de um período bastante longo de tempo [47].

Para o processo de medição existem várias alternativas, como por exemplo as aplicações *Iperf*, MGEN [48] e D-ITG [49]. A escolha recai sobre o *Iperf*, devido a dois factores: a inclusão prévia desta ferramenta na OML e o facto de fornecer várias métricas, para além da taxa de transferência. Todas estas ferramentas operam de forma semelhante: através de dois módulos, cliente e servidor, são gerados dados no sentido do cliente para o servidor, permitindo à aplicação a medida do tempo tomado entre nós, efectuando de seguida o cálculo tendo em conta a quantidade de informação gerada.

3.1.3 Perda de Pacotes

Ao longo de uma rota os pacotes emitidos pelo remetente podem-se perder, não chegando a ser recebidos pelo destino. Regra geral, a perda de um único pacote não possui um impacto relevante na aplicação. Porém, a perda repetida de pacotes pode ter um efeito significativo. A perda de pacotes pode ser influenciada por factores como a qualidade da ligação, os nós de origem e de destino e os protocolos de rede utilizados [50].

O estudo desta métrica é importante por diferentes razões: algumas aplicações têm dificuldades em lidar com grandes quantidades de pacotes perdidos, podendo até interferir com o seu desempenho; aplicações com necessidades de tempo real não suportam perdas excessivas; quanto maior a taxa de perda de pacotes menor será a taxa de transferência, porque, como visto na secção anterior, a taxa de transferência depende do sucesso da entrega de pacotes [51].

A medição da taxa de perda de pacotes é feita através da geração de tráfego e uma posterior contagem do número de pacotes que chegam ao destino. Neste trabalho recorreu-se novamente à aplicação *Iperf* pelas razões enunciadas anteriormente. É ainda possível fazer esta medição com um simples comando *ping*. Todavia não é uma opção viável, por não ser possível

obter perdas que correspondam às reais, devido aos pacotes ICMP terem um tamanho por defeito menor e serem transmitidos de forma regular, em oposição à transmissão geralmente mais irregular das aplicações de rede.

3.1.4 Atraso

Uma das definições geralmente aceites do atraso é o tempo que passa desde que o primeiro bit de um pacote passa num primeiro ponto de observação da rede (e.g. interface de um nó), até à passagem do último bit desse mesmo pacote no segundo ponto de observação da rede (que pode coincidir com o primeiro), medido em segundos [52] [53].

A análise desta métrica é importante por variadas razões: à semelhança da taxa de pacotes perdidos algumas aplicações têm dificuldade em operar num ambiente com atrasos demasiado grandes; variações irregulares no atraso tornam o suporte a aplicações de tempo real difícil, ou até impossível; quanto maior for o valor de atraso menor é o valor da taxa de transferência; o valor mínimo desta métrica corresponde ao valor do atraso devido à propagação e de transmissão; valores acima deste valor mínimo são uma indicação do congestionamento da rede [54].

O processo de medida do atraso baseia-se na aplicação de uma marca de tempo [55], que consiste numa sequência de informação a denotar a data e/ou hora a que um determinado evento ocorre (por exemplo *Sat Jul 23 02:16:57 2005*), nos pacotes no nó de origem e ao chegarem ao receptor é comparada essa marca com o tempo de chegada (este processo implica uma sincronização entre os relógios dos nós). A Figura 8 demonstra um exemplo bastante simples da medição do atraso entre dois nós: o primeiro nó, ao enviar os pacotes de teste aplica-lhes uma marca de tempo com o valor de 0 segundos e à chegada ao nó receptor é comparado o tempo de chegada, 17ms, com o valor da marca de tempo, e efectuado o cálculo desta métrica, encontrando um atraso de 17ms.

Figura 8 - Exemplo de medição do atraso

Esta métrica é possível de obter através do comando *ping* que é bastante utilizado com este propósito [50]. Inclusive, na documentação da biblioteca de medidas é aconselhado o recurso ao ping para obtenção de valores de atraso [56].

No entanto, não se trata de um processo de medida implementado na OML, através do qual os resultados sejam armazenados na base de dados, o que leva a que apesar de possível de realizar uma experiência para obter valores de atraso não será uma métrica sob a qual este trabalho se centra.

3.1.5 *Jitter*

Esta é uma métrica alvo de alguma discórdia na sua definição [39].

Por um lado, segundo [33], dado um fluxo, de pelo menos dois pacotes, que atravesse um ponto de observação A e um segundo ponto de observação B, *jitter* (ou variação no atraso de pacotes IP) é definido como a diferença de atraso entre um par de pacotes desse fluxo.

Por outro lado [57], ITU-T [58] propõe uma métrica ligeiramente diferente da proposta pelo IETF [59]: *jitter* é a diferença entre 99,9% dos atrasos registados e um atraso referência, preferencialmente o menor atraso. De notar que numa situação em que o menor atraso seja o do pacote anterior estas definições coincidem.

Esta métrica, cuja unidade é o segundo, pode ser medida utilizando mais uma vez a aplicação *Iperf*. De acordo com [60], a definição de *jitter* adoptada pelo *Iperf* é a do IETF, estando a sua formulação em [61]. O processo de medida de *jitter* passa pela utilização de uma sequência de pacotes pré-definida entre dois pontos de observação, em que o nó de recepção, servidor do *Iperf*, tem conhecimento das características do tráfego (e.g. taxa de pacotes, distribuição do tamanho dos pacotes, intervalo entre pacotes) gerado pelo cliente, sendo depois possível com essa informação aplicar uma marca de tempo nos pacotes recebidos e determinar a variação no atraso entre cada par de pacotes.

3.2 Métricas Complexas

3.2.1 Expected Transmission Count (ETX)

Actualmente, a métrica mais utilizada por protocolos de encaminhamento em redes ad hoc é o número mínimo de saltos [40]. Esta abordagem assume implicitamente que as ligações ou trabalham ou não trabalham, o que não é verdade para ligações sem fios onde existem rácios de perdas de pacotes intermédios. Minimizar o número de saltos implica aumentar a distância de cada salto (Figura 9) o que leva a uma provável diminuição de força do sinal, maximizando a taxa de perda de pacotes [13].

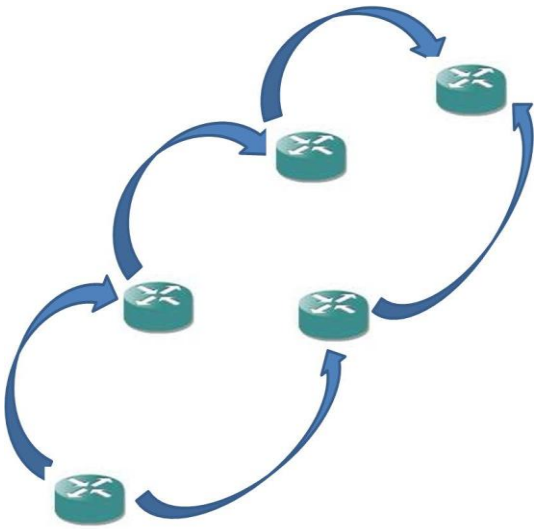

Figura 9 - Percursos alternativos com 2 e 3 saltos

É neste contexto que surge a métrica ETX, proposta e avaliada em [13], com o objectivo principal de calcular as rotas com a taxa de transferência mais alta, de ponta a ponta. Para isso, a estimativa do ETX tem em consideração o intervalo de valores que os rácios de perdas podem assumir, a existência de ligações com taxas de perdas assimétricas e a interferência entre saltos sucessivos em redes com múltiplos saltos.

Esta métrica é calculada usando os rácios de entrega de pacotes no sentido remetente para destinatário e no inverso:

$$ETX = \frac{1}{d_f \times d_r} \qquad (1)$$

com d_f a probabilidade de um pacote ser entregue com sucesso ao seu destinatário e d_r a probabilidade do pacote de resposta enviado pelo nó de destinado ser recebido com sucesso pelo nó origem inicial.

O ETX possui um conjunto de características importantes: o facto de ser baseado em rácios de entrega, que afectam directamente a taxa de transferência; detecta e lida com assimetrias na ligação ao incorporar as taxas de perdas em ambas as direcções; penaliza rotas com maior número de saltos, que possuem menor taxa de transferência devido à interferência entre diferentes saltos da mesma rota; tende a minimizar a utilização do espectro, o que deve maximizar a capacidade total do sistema.

3.2.2 Expected Transmission Time (ETT)

Esta métrica pode ser definida como uma extensão da métrica anterior, ETX, para ter em conta a largura de banda.

A métrica ETT estima o tempo que um pacote de dados leva a ser transmitido com sucesso, colmatando duas lacunas deixadas pelo ETX, a distinção entre ligações com diferentes capacidades e o facto da probabilidade de perda de pequenos pacotes de teste ser diferente da de pacotes de dados [62].

A fórmula para o seu cálculo é portanto o resultado do apontado no parágrafo anterior, o produto de ETX com o tamanho do pacote de teste, dividido pela largura de banda:

$$ETT = ETX \times \frac{S}{B} \qquad (2)$$

sendo S o tamanho dos pacotes e B a largura de banda da ligação [63].

Para um uso eficaz desta métrica é recomendado o uso de pacotes de teste direccionados para todos os vizinhos, o que apesar de permitir obter resultados mais precisos, é uma abordagem com problemas a nível de escalabilidade, em especial em redes mais densas [64].

3.2.3 Weighted Cumulative Expected Transmission Time (WCETT)

WCETT é uma métrica baseada no ETT, e implicitamente no ETX. Esta métrica acrescenta às anteriores o facto de também contabilizar a diversidade de canais e a interferência entre ligações que utilizem o mesmo canal [63].

Esta métrica parte de algumas premissas: todos os nós na rede são estacionários, o que invalida a sua utilização num futuro cenário de mobilidade; nós equipados com mais do que uma placa de rede 802.11 devem tê-las a operar em diferentes canais, que não causem interferências mútuas [65].

O cálculo do WCETT pode ser interpretado como uma sucessão de passos que acrescentam aspectos tidos em conta nesta métrica [41]. Numa primeira fase define-se WCETT em função de ETT, combinando os ETTs das ligações ao longo de uma rota, obtendo-se uma estimativa do atraso ponta a ponta:

$$WCETT = \sum_{i=1}^{n} ETT_i \qquad (3)$$

em que ETT_i é a métrica ETT para a ligação i da rota e n o número de saltos da rota.

É de seguida necessário considerar o impacto da diversidade de canais. Considerando um total de k canais no sistema, define-se Xj como:

$$Xj = \sum_{i=1}^{n} ETT_i \,; 1 \leq j \leq k \qquad (4)$$

sendo assim Xj é a soma dos tempos de transmissão no canal j. O canal dominante da rota será o que tem maior Xj. É possível de simplificar (4) da seguinte forma:

$$WCETT = \underset{1 \leq j \leq k}{max\ Xj} \qquad (5)$$

Combinando (3) e (5) obtém-se então a fórmula final para WCETT:

$$WCETT = (1 - \beta) \times \sum_{i=1}^{n} ETT_i + \beta \times \underset{1 \leq j \leq k}{max\ Xj} \qquad (6)$$

em que β é um parâmetro ajustável, entre 0 e 1.

A equação (6) pode ser interpretada como uma média ponderada, de forma a obter um balanço entre os dois factores da equação, entre um primeiro termo, que é a soma dos tempos de transmissão ao longo da rota, e um segundo termo, que corresponde ao conjunto de saltos que terão maior impacto.

Em suma, WCETT é uma estimativa com os seguintes objectivos: considerar as taxas de perdas como a taxa de transferência, aumentar ao longo da rota, e ter em conta a redução da taxa de transferência devido à interferência entre ligações a operar no mesmo canal. De acordo com [63] os seus resultados são 250% mais eficazes que o uso do número de saltos no cálculo de rotas e 80% melhor que o ETX.

Capítulo 4 - Ferramentas

Após o estudo das métricas para análise de desempenho de redes procede-se ao estudo das possibilidades existentes para os vários componentes da solução proposta.

A visão geral da solução, representada na Figura 10, demonstra os três componentes base: uma ferramenta de geração de tráfego, a laranja; uma plataforma de recolha de medidas, a azul; e ferramentas para lidar com o protocolo de transporte de fluxos, a verde.

Neste capítulo são analisados em pormenor a plataforma de gestão e controlo OMF e a biblioteca de medidas OML, visto serem as plataformas base deste trabalho. Nas secções seguintes serão estudadas soluções para a geração de tráfego artificial, a partir do qual serão recolhidas as métricas pretendidas. E por fim são analisadas as possibilidades para os protocolos de transporte de fluxos e as ferramentas para a sua exportação e recolha.

A conjugação destes três módulos, OML, uma ferramenta de geração de tráfego e protocolo de transporte de fluxos dão origem à solução proposta neste trabalho, como se pode ver, de uma forma generalizada, na Figura 10.

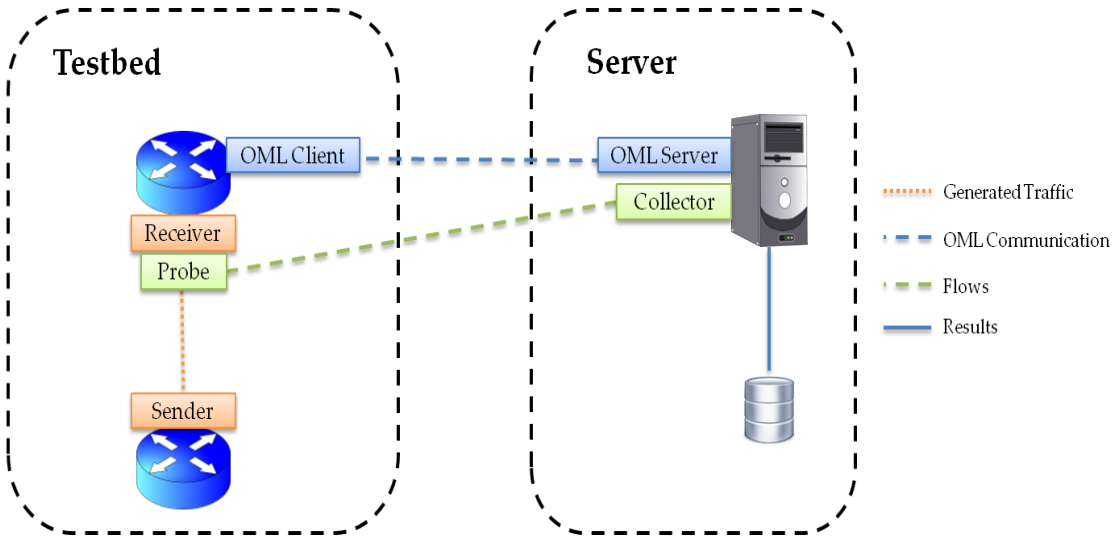

Figura 10 - Visão Geral da Solução Proposta

4.1 OMF (*cOntrol and Management Framework*)

Esta plataforma foi, como referido na secção 2.3, inicialmente criada para a rede de testes *Orbit*. Desde 2007 que tem sido alargada a outras redes, com o objectivo de uniformizar o processo de experimentação, com diferentes tipos de tecnologias de redes e recursos.

Esta plataforma existe com duas finalidades [5], descritas na Tabela 2.

OMF (*cOntrol and Management Framework*)			
Utilização da Rede		**Gestão da Rede**	
Validação, precisão e reprodutibilidade	Suporte de "ciclos de experiências" e rigor científico	Facilidade de operação e manutenção	Optimização da utilização dos recursos da rede

Tabela 2 - OMF

A OMF é uma plataforma genérica para controlo e gestão de redes experimentais, fornecendo várias funcionalidades ao utilizador:

- Uma linguagem de programação específica, designada OEDL, que permite:
  - Descrever os recursos necessários para a experiência e a sua configuração;
  - Descrever as aplicações a usar e as medidas a recolher;
  - Descrever as tarefas a executar durante a experiência e a sua calendarização.
- Um conjunto de ferramentas de *software* que:
  - Aceitam como entrada a experiência anteriormente descrita;
  - Inicializam e configuram os recursos e aplicações necessários;
  - Enviam os comandos aos recursos para executarem as tarefas planeadas no momento certo;
  - Recolhem os dados das medidas;
  - Acedem e analisam os resultados experimentais.

A Figura 11 [66] fornece uma visão global desta plataforma: na perspectiva do utilizador, cabe-lhe a criação da descrição da experiência (ED) e, no final da experiência a recolha de resultados. O decorrer da experiência passa-se de forma alheia ao utilizador: a plataforma OMF, com base na ED configura a rede (aplicações, recursos, etc.) e dá início à experiência, fazendo as medições pedidas pelo utilizador, disponibilizando-lhe esses resultados no final.

Uma descrição mais detalhada do processo de execução de uma experiência pode ser observada na Figura 12 [66].

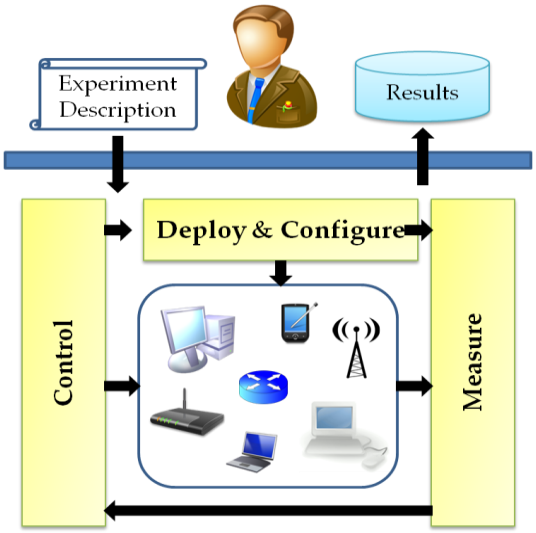

Figura 11 - Visão geral OMF

Assumindo um utilizador chamado Alice, este processo pode ser definido através dos seguintes passos:

- O utilizador "Alice" começa uma nova instância de um Controlador de Experiência (EC) e passa-lhe a sua Descrição da Experiência (ED)
- O EC interpreta a ED e envia ao Gestor do Agregado (AM) da rede os pedidos de inicialização e configuração dos recursos definidos por Alice
- Os múltiplos serviços do AM têm início e configuram os recursos
- Quando os recursos estiverem preparados, o EC envia os comandos ao Controlador de Recursos (RC), que está a correr em cada um desses recursos
- Os RCs interpretam esses comandos e executam as acções, tais como arrancar a aplicação M, alterar os parâmetros de N, etc.
- Nesta fase a experiência está a decorrer
- Enquanto a experiência está a correr as aplicações podem enviar dados das medidas para a Biblioteca de Medidas (ML)
- A ML pode fazer algum pré-processamento de filtragem dos resultados
- A ML envia os dados pré-processados para o Serviço Colector de Medidas, que os armazena numa base de dados exclusiva desta experiência

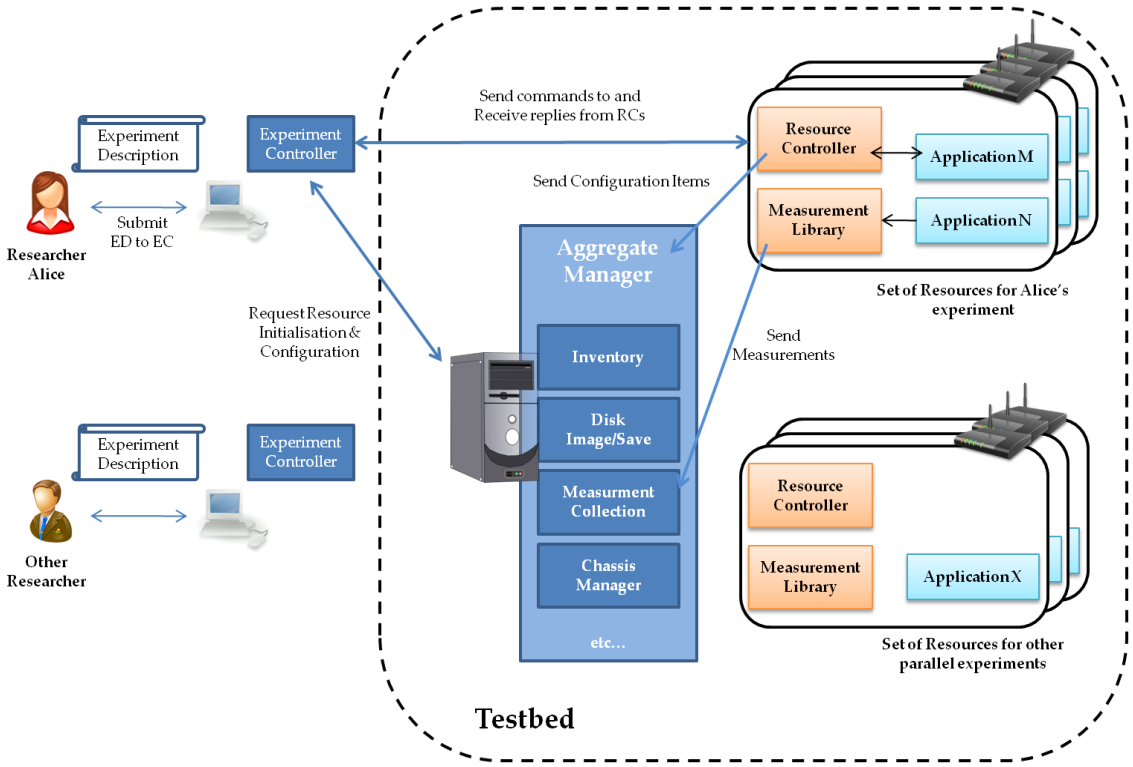

Figura 12 - Arquitectura do sistema OMF

É ainda perceptível através da Figura 12 a possibilidade de haver várias experiências a decorrer em simultâneo, desde que cada experiência use um conjunto de recursos disjuntos, i.e. o mesmo recurso (e.g. nó) não pode ser partilhado por experiências concorrentes.

É no âmbito do projecto OMF que surge a biblioteca de medidas OML, para lidar com todo o processo de medição e processamento de resultados, como veremos de seguida.

4.2 OML (OMF *Measurement Library*)

Esta biblioteca de medidas é a base para todo o trabalho desenvolvido nesta dissertação: é a partir da OML que será desenvolvida a solução proposta, estendendo-a com suporte IPFIX.

Apesar de criada como parte do projecto OMF, não existe uma inter-dependência entre as plataformas, ou seja, a biblioteca OML pode ser utilizada sem a OMF, e vice-versa. Na Figura 13 [66] está patente a interacção da OML com a OMF.

A OML possibilita a recolha de dados em tempo real das várias aplicações executadas durante a experiência, fornecendo ainda uma Biblioteca de Medidas, que permite ao utilizador definir pontos de medida no código fonte da sua aplicação (C/C++). É também possível aos utilizadores definir e configurar filtros para um pré-processamento à informação recolhida dos pontos de medida. Em alternativa, caso não esteja disponível o código fonte da aplicação, com a

OML é possível criar um *script* que envolva a aplicação de maneira a recolher os seus parâmetros de saída como medidas.

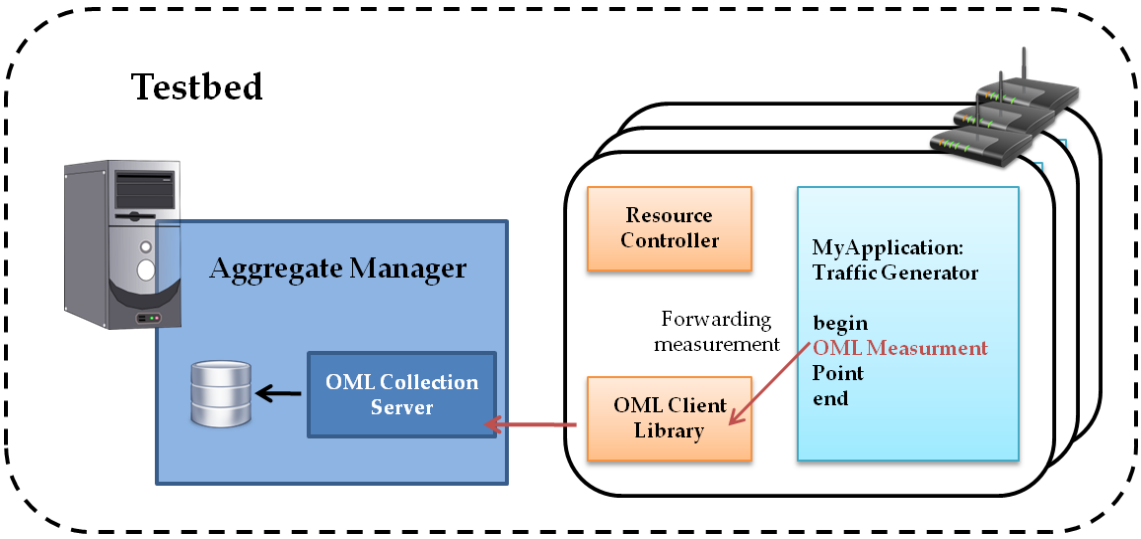

Figura 13 - Interacção OML – OMF

A OML é baseada numa arquitectura cliente/servidor e é constituída por um Servidor de Colecção OML e uma Biblioteca de Medidas OML (cliente). O Servidor de Colecção corre como um *daemon* no servidor da rede experimental e recolhe os resultados dos vários recursos especificados na experiência. Cada um desses recursos, por sua vez, possui um cliente OML, que recebe as medições das aplicações, aplicando-lhes os filtros de pré-processamento definidos e reencaminhando a informação para o Servidor de Colecção OML.

Uma vez concluída a experiência, os resultados são armazenados numa base de dados SQLite [67] e podem ser acedidos para posteriores análises, através de *queries* SQL ou exportando-os para futuro processamento. Vejamos as opções de consulta de dados:

- Utilizar os serviços disponibilizados pelo Gestor do Agregado
    - É o método usado por defeito e o recomendado para aceder às medidas. Fazendo uso de um Web browser é possível aceder à base de dados.
- Utilizar *SQLite* directamente
    - Desta maneira acede-se à base de dados através da manipulação directa com o *SQLite*, com *queries* SQL.
- Exportar a base de dados para um ficheiro de texto
    - É possível exportar o conteúdo das tabelas para um ficheiro de texto, podendo depois ser facilmente importado para as ferramentas tradicionais de processamento de resultados, como por exemplo Matlab [68] e Excel [69].

Tendo em conta que nesta dissertação é usada a plataforma OMF para gestão da rede de testes, é interessante fazer uma análise mais aprofundada do processo de experimentação na OMF, com recurso à OML. A Figura 14 [70] demonstra esquematicamente este processo.

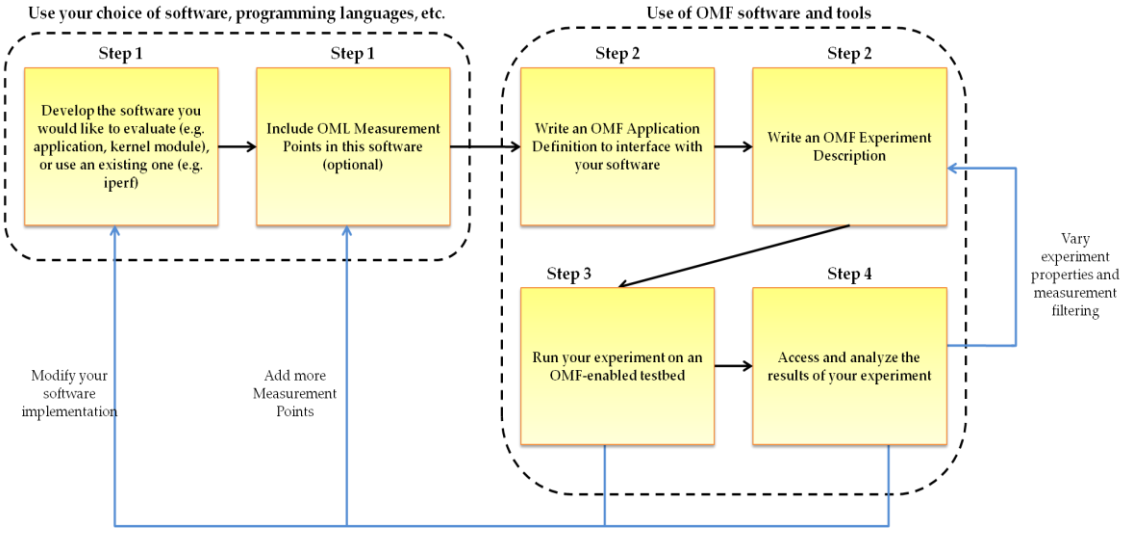

Figura 14 - Visão geral da utilização da OMF com OML

O primeiro passo é o desenvolvimento do software a ser avaliado, com quaisquer ferramentas de programação e linguagem, desde que o resultado final seja um binário executável no sistema operativo instalado nos recursos.

De seguida, é necessário criar a experiência pretendida: a definição da aplicação OMF (AD) para a aplicação que se pretende testar, funcionando como uma interface que torna as entidades OMF conscientes da aplicação do utilizador, i.e. que parâmetros aceita, que pontos de medida fornece, etc.; depois há que escrever uma descrição da experiência OMF (ED), em OMF *Experiment Description Language* (OEDL), que como o próprio nome indica é a linguagem específica da OMF para descrição de experiências.

O terceiro passo abrange vários aspectos, como obter uma conta numa rede experimental com a plataforma OMF, fazer a reserva dos recursos necessários, aceder à consola da rede durante o período reservado, enviar a ED e correr a experiência.

Por fim, resta aceder aos resultados obtidos e analisá-los das formas anteriormente abordadas.

Como se pode ver a OML é uma plataforma para recolha de medidas, i.e. a OML oferece uma forma de vários dispositivos, ligados em rede, poderem enviar os dados, até então guardados localmente, para uma estação central, onde são salvaguardados. Outra vantagem desta aplicação é o facto de ser bastante flexível, possibilitando recolha de todo o tipo de dados estatísticos, desde utilização de CPU a localizações GPS.

Contudo, a OML possui algumas limitações, uma das quais é objectivo desta dissertação ultrapassar: a necessidade de, como verificado nos passos a tomar na realização de uma experiência, a criação da AD, ou seja, um utilizador que pretenda utilizar uma aplicação nesta plataforma tem que passar primeiro por um processo de instrumentação da mesma. Esta instrumentação significa adicionar pontos de medida OML à aplicação, para que a OML consiga recolher os dados e armazená-los. Para o fazer existem dois cenários: no primeiro o código fonte da aplicação está disponível em C/C++ ou no segundo caso não se encontra disponível o código fonte, mas sim o binário executável.

No primeiro caso, em que se tem disponível o código fonte em C/C++, a inclusão de pontos de medida no código fonte passa por adicionar algumas linhas de código, como descrito no Apêndice A desta dissertação.

A segunda opção, para quando o código da aplicação não se encontra disponível em C/C++, consiste no desenvolvimento de um *script* para a aplicação, que a iniciará, capturará os seus resultados e passa-os para a OML, usando a linguagem de programação Ruby [71].

4.3 Ferramentas de Geração de Tráfego

Como se pode observar na Figura 10 é agora necessário avaliar as possibilidades para ferramentas de geração de tráfego, a integrar na OML, da qual serão exportadas as métricas por fluxos. A escolha recaiu sobre o *Iperf* por permitir a recolha de métricas interessantes (taxa de transferência, *jitter* e taxa de pacotes perdidos) e por ser a única ferramenta geradora de tráfego já plenamente implementada na OML, o que facilita não só a sua integração na solução proposta, mas também a posterior comparação entre os resultados obtidos através da OML, e através do processo proposto neste trabalho.

Existe uma panóplia de aplicações para o efeito, das quais se distinguem *Iperf*, MGEN e D-ITG, pela facilidade de implementação, operação e por providenciarem os resultados pretendidos, duma forma simples e acessível. Por serem as aplicações mais interessantes do ponto de vista deste projecto serão alvo de um estudo mais detalhado nas secções seguintes.

Para além destas duas opções foram ainda testadas várias outras alternativas, tendo sido descartadas pelas mais variadas razões (processo de análise de resultados incompleto, sistema operativo inadequado, resultados insuficientes ou dificuldades de instalação ou execução). Essas ferramentas encontram-se descritas na Tabela 3.

Aplicação	Descrição	Referências
RUDE & CRUDE	RUDE é uma ferramenta de geração de tráfego UDP, que é recebido e registado pelo colector CRUDE.	[72]
UDPmon	UDPmon é uma aplicação criada para testar a rede através do envio de pacotes UDP.	[73]
Netperf	*Netperf* é uma ferramenta semelhante ao *Iperf*, com o objectivo de analisar o desempenho da rede.	[29]
TfGen	*TfGen* é um gerador de tráfego TCP, sem quaisquer funções de análise.	[74]
KUTE	KUTE é uma aplicação idêntica ao RUDE & CRUDE, com uma componente para a geração de tráfego UDP e outra para a recepção.	[75]
UDP Generator	Ferramenta utilizada para testes na rede, dividida num módulo de geração de pacotes UDP e um para recepção.	[76]

TrafGen	TrafGen gera e recebe vários tipos de tráfego (UDP, TCP, RTP, HTTP) para avaliar o desempenho da rede.	[77]
Traffic Generator Tool	Aplicação para a geração de tráfego TCP e UDP, sem capacidade para análise dos resultados obtidos.	[78]

Tabela 3 - Ferramentas de Geração de Tráfego

4.3.1 *Iperf*

Iperf é uma das ferramentas mais comuns para testes de fluxos TCP [79] e UDP, sendo possível obter resultados para a taxa de transferência, *jitter* e perda de pacotes.

Trata-se de uma aplicação com duas vertentes, servidor e cliente, que funcionam em complementaridade: cabe ao cliente gerar o tráfego que será recebido pelo servidor, permitindo assim as medições referidas no parágrafo anterior.

Este software possui ainda a vantagem de ser possível configurar vários parâmetros, tais como o tipo de tráfego gerado, o tamanho das mensagens enviadas ou a duração da experiência.

A título de exemplo, para correr uma experiência com tráfego UDP, de duração de 60 segundos, com relatórios das estatísticas de 5 em 5 segundos foram utilizadas as seguintes instruções:

- Do lado do servidor: *iperf –s –u –i 5*

- Do lado do cliente: *iperf –c 192.168.0.3 –u –i 5 –t 60*

Obtendo assim os resultados apresentados no Apêndice B.

Como referido no início da secção 4.3, o *Iperf* encontra-se integrado na plataforma de medidas OML [56]. Este factor foi decisivo na escolha da ferramenta geradora de tráfego. O facto de já estar incluído na plataforma é sinónimo de garantia de uma compatibilidade completa com o processo de recolha de medidas da OML tornando possível uma comparação entre resultados, dado que o gerador de tráfego será comum à OML e ao sistema implementado.

4.3.2 *Multi-Generator* (MGEN)

Este projecto, actualmente na versão 5.01 é um *software open source* [80] para realização de testes de desempenho de redes IP e medidas usando tráfego UDP.

Este conjunto de ferramentas permite a geração de padrões de tráfego em tempo real, sendo depois o tráfego registado e analisado. A definição dos padrões de geração de tráfego, que podem emular aplicações UDP *unicast* e/ou *multicast*, é feita através de *scripts* criados pelo utilizador. Posteriormente, a análise dos registos do MGEN permite fazer cálculos de desempenho da rede, e.g. taxa de transferência, perda de pacotes ou atraso [81].

O Apêndice C possui o exemplo de uma experiência MGEN. O *script* leva à geração de quatro fluxos de tráfego UDP, cujos inícios distam 15 segundos entre si, finalizando ao fim de 60

segundos, desde o começo da experiência. Analisando o primeiro fluxo pode-se verificar que o tráfego tem origem no nó em que é corrido o script, na porta 5001, tendo como destino a porta 5000 do endereço IP 10.110.1.2, com uma distribuição periódica, enviando mensagens de 4096 bytes, a um ritmo de 10.000 por segundo.

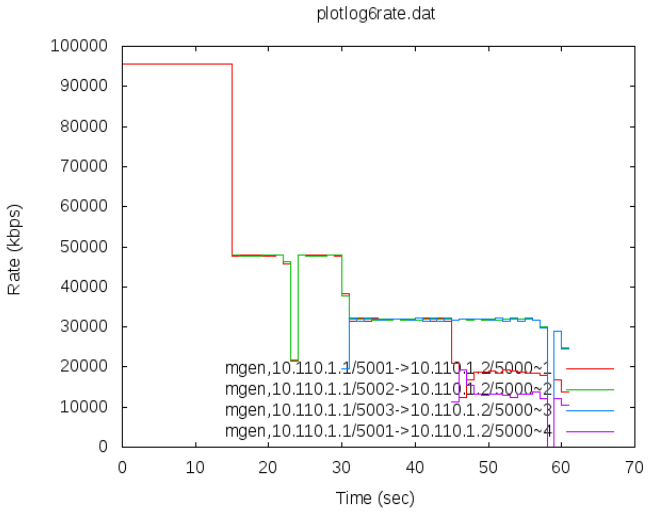

Figura 15 - Exemplo de resultados para a taxa de transferência com MGEN

Por fim, é possível através da análise do ficheiro com os dados estatísticos obter vários gráficos, como é exemplo a Figura 15 que representa os resultados para a taxa de transferência da experiência descrita no Apêndice C. No *script* utilizado para definição da experiência foram definidos quatro fluxos UDP gerados com intervalos de 15 segundos, que correspondem às diferentes cores na Figura 15. A primeira cor corresponde ao tráfego gerado a partir do início da experiência, atingindo o valor máximo de taxa de transferência (aproximadamente 96 Mbps), valor este que decresce conforme é iniciada a geração de tráfego dos restantes fluxos definidos, representada pela introdução de uma nova cor no gráfico a cada 15 segundos. A experiência cessa 60 segundos após o seu começo, como definido no *script* do Apêndice C.

4.3.3 *Distributed Internet Traffic Generator* (D-ITG)

Esta ferramenta, projecto da Universidade *Federico II* de Nápoles, é uma plataforma capaz de produzir tráfego, bastante fiel aos processos estocásticos para as variáveis aleatórias (exponencial, uniforme, cauchy, etc.) de IDT e PS. O D-ITG funciona em IPv4 e IPv6 e consegue gerar tráfego ao nível das camadas de rede, transporte e aplicações. Os protocolos suportados são TCP, UDP, ICMP, DNS, Telnet e VoIP.

As métricas passíveis de serem retiradas através desta plataforma são: atraso num sentido, RTT, pacotes perdidos, *jitter* e taxa de transferência. Estas informações podem ser armazenadas tanto no nó remetente como no de destino. É ainda possível definir a duração e atraso (em relação ao momento inicial) da experiência [82].

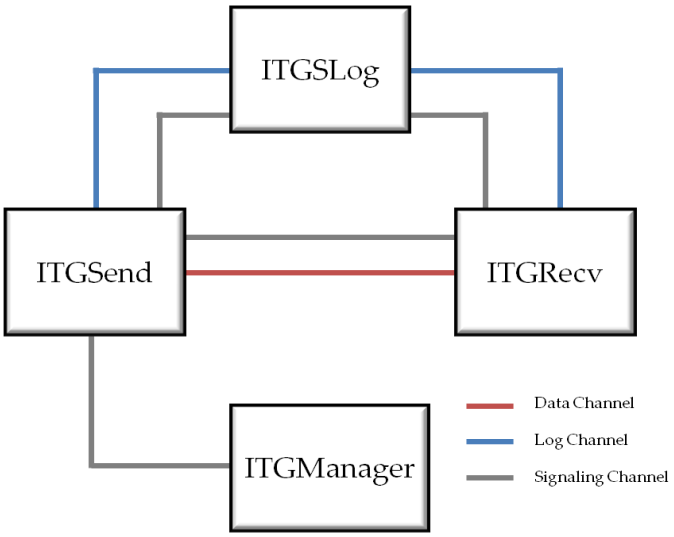

Figura 16 - Arquitectura D-ITG

Na Figura 16 [83] tem-se uma visão geral do sistema de controlo do D-ITG: o módulo *ITGSend*, a correr no remetente onde são feitas as configurações dos fluxos a enviar; *ITGRecv*, no destinatário a aguardar o tráfego gerado; *ITGManager*, para controlo remoto do *ITGSend*; *ITGLog*, que actua como um servidor de registo remoto. Para além destes componentes ainda existe o *ITGDec*, que faz o tratamento da informação registada, dando como resultado as métricas desejadas.

É possível consultar um exemplo de uma experiência simples, utilizando o D-ITG no Apêndice D.

4.4 Protocolos de Transporte de Fluxos

Numa época em que as redes sem fios estão em constante expansão, quer em tamanho quer em capacidade, trabalhar com todos os dados na rede torna-se uma tarefa penosa a nível de processamento, memória e largura de banda.

A solução para a análise destas redes consiste na selecção de algumas estatísticas, que descrevem os pacotes e posterior agregação desta informação em fluxos [84].

Existem diferentes soluções para a exportação destes fluxos para um servidor central onde são armazenados e analisados, como se pode constatar nas secções seguintes.

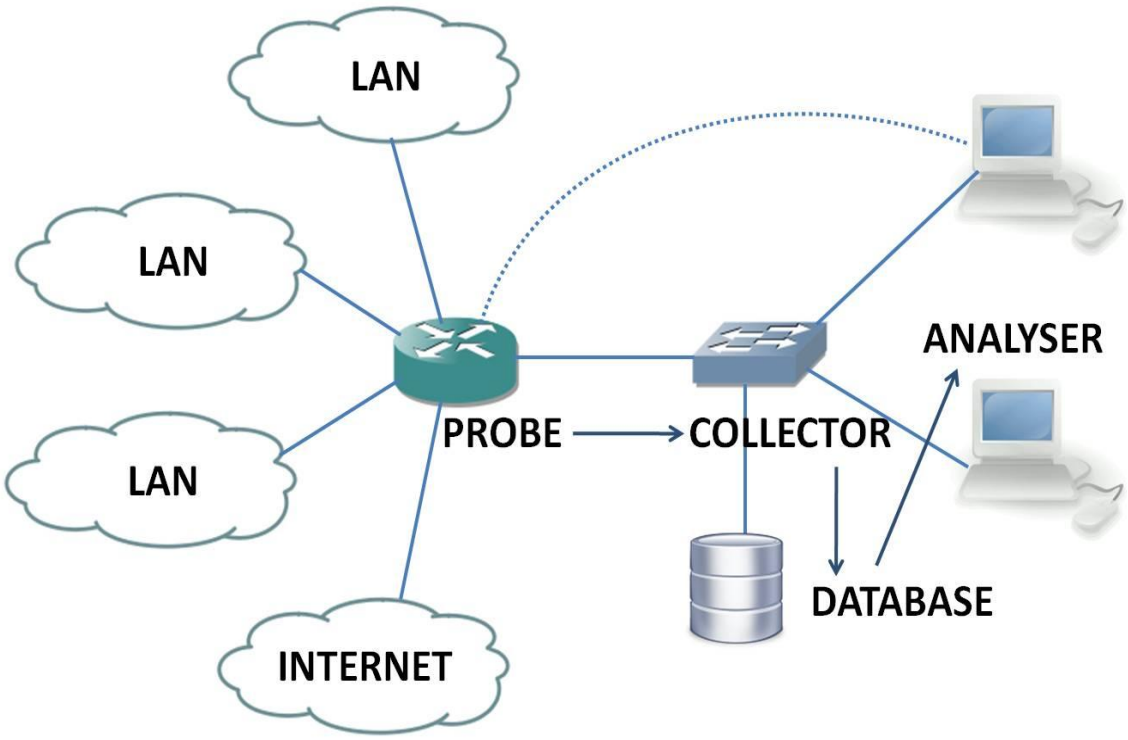

Figura 17 - Visão geral da exportação de fluxos

De uma forma genérica (Figura 17) o processo de monitorização de tráfego através de fluxos é iniciado nas *probes* (software de exportação instalado nos nós) através da exportação de fluxos de tráfego IP para um colector, para serem depois guardados e usados em posteriores análises.

4.4.1 sFlow

O protocolo *sFlow* [85] é a primeira solução estudada nesta dissertação para a exportação dos fluxos que contêm a informação dos pacotes na rede.

A norma *sFlow* utiliza uma tecnologia de amostragem para capturar estatísticas da rede enviando-as depois para um servidor central [86]. O *sFlow* pode ser dividido em dois módulos (ver Figura 18 [87]):

- Agente *sFlow*: implementação do mecanismo de amostragem nos nós.
- Colector *sFlow*: servidor central que recolhe os datagramas sFlow enviados pelos agentes.

O sistema de monitorização *sFlow* foi criado com o objectivo de permitir uma monitorização precisa do tráfego na rede, mesmo a velocidades elevadas; escalabilidade, para permitir monitorizar milhares de agentes com um único colector; implementação acessível, na medida em que pode ser instalado sem requisitos adicionais de memória e CPU [88] [89].

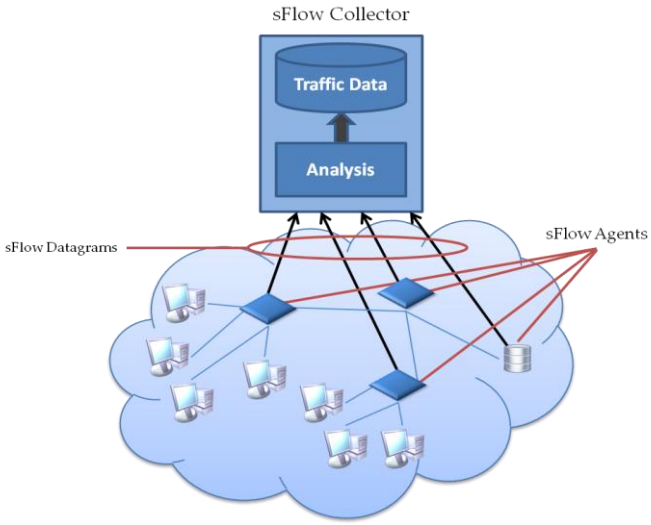

Figura 18 – sFlow

O *sFlow* é uma tecnologia de amostragem que possibilita uma monitorização contínua da rede da seguinte forma: os agentes *sFlow* são um software que corre nos nós, encapsulando informação das amostras capturadas, enviando-a de imediato para o colector, necessitando assim de um esforço computacional e de memória mínimo. Como se pode constatar na Figura 18 o funcionamento do sistema *sFlow* é o seguinte: os agentes dispostos pela rede estão continuamente a enviar datagramas *sFlow* para o colector, fornecendo uma visão em tempo-real, rica em informação sobre o fluxo de dados na rede.

Este processo resulta numa monitorização da rede precisa, detalhada, escalável e de baixo custo, cuja amostragem é feita ao nível do hardware, suportando cargas elevadas com possibilidade de quantificar os erros [87].

4.4.2 NetFlow

Netflow é a solução da *Cisco Systems* [90] para recolha de informações sobre o tráfego na rede. A *Cisco* define o *NetFlow* como um conjunto de serviços para aplicações IP, incluindo contabilidade de tráfego na rede, planeamento de rede, segurança e monitorização. NetFlow fornece informação importante sobre os utilizadores da rede, as aplicações, picos de tempo de utilização, e o encaminhamento do tráfego [91].

Este protocolo já conta com nove versões, sendo a versão 9 a mais recente e mais generalizada. As versões anteriores tiveram bastante aceitação, em especial a versão 5, sendo largamente utilizada para exportação e recolha de informação de fluxos IP [92].

Netflow utiliza na realidade um protocolo bastante simples, no qual o exportador envia a informação, *template* e opções do fluxo para o colector. São considerados fluxos sequências unidireccionais de registos de dados com formatos semelhantes. Tipicamente, no caso de fluxos

Netflow, podem ser identificados através de sete campos chave: endereço IP da origem, endereço IP de destino, porta de origem, porta de destino, tipo de protocolo da camada 3 (por exemplo IP e ICMP), ToS [93] e *interface* de saída. Os fluxos *Netflow* transportam informação como endereços IP, contagem de pacotes e *bytes*, marcas de tempo, portas utilizadas, etc. [94].

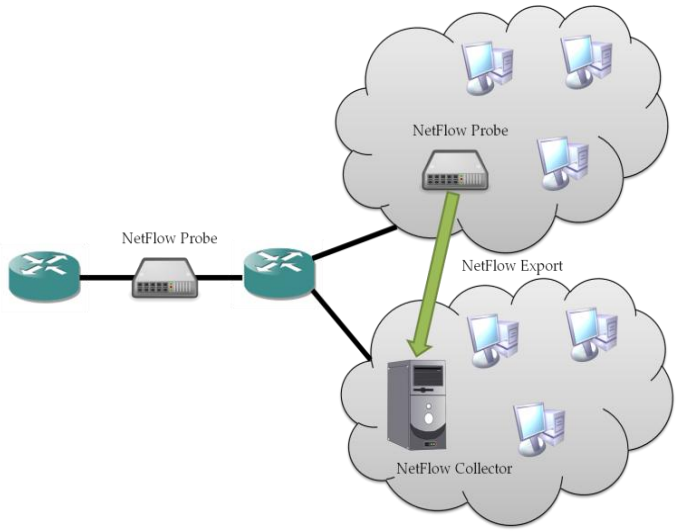

Figura 19 - Arquitectura NetFlow

Na Figura 19 [95] pode-se constatar a aplicação do protocolo *NetFlow* numa rede: são instaladas as *probes* em pontos da rede que se pretendam avaliar, e estas enviam os fluxos para o colector que os armazenará [96].

Este protocolo deu origem ao IPFIX, protocolo estudado na secção seguinte.

4.4.3 *IPFIX*

Internet Protocol Flow Information eXport (IPFIX) [97] é a tentativa do IETF em constituir um protocolo normalizado para a exportação de fluxos com informação relativa a medições efectuadas em tráfego IP na rede, definido em [98], correspondendo aos requisitos expostos em [99]. Como mencionado previamente, IPFIX é baseado em *Netflow* versão 9, por várias razões [92], entre as quais o facto da possibilidade de descrição de templates, conferindo uma maior flexibilidade ao protocolo, e por tal partilha algumas características com a norma proprietária da *Cisco*.

Ao comparar a Figura 19 e a Figura 20 [100] chega-se à conclusão que possuem arquitecturas equivalentes: são colocados exportadores nos nós que enviam a informação medida para o colector, duma forma *push*, i.e. os exportadores estão periodicamente a enviar fluxos, sem qualquer pedido do colector, sendo portanto da responsabilidade dos exportadores dar início ao processo [101].

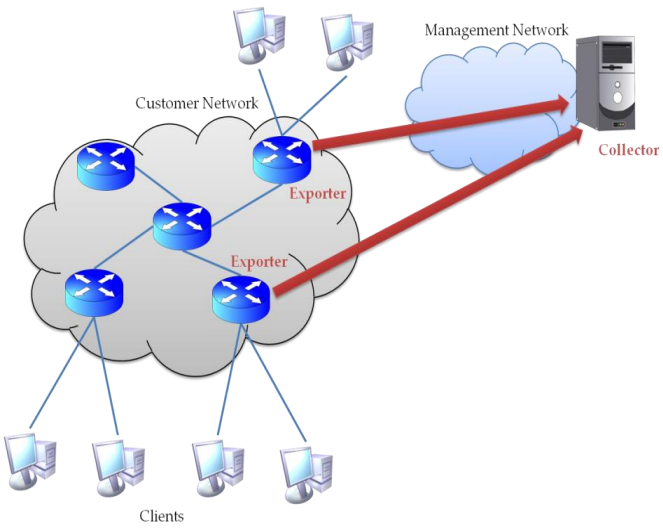

Figura 20 – IPFIX

No IPFIX a relação entre exportadores e colectores é de muitos para muitos, ou seja, um exportador pode reportar para vários colectores, enquanto um colector pode receber informação de vários exportadores [102].

Neste trabalhou optou-se por implementar IPFIX, por diferentes razões, das quais importa salientar que se trata de um protocolo criado com o objectivo de se tornar no padrão para a exportação de fluxos IP, o facto de ser uma evolução do *Netflow*, permitir uma flexibilidade superior à do *sFlow*, através da definição de *templates* [103] e uma maior simplicidade na implementação de um colector [92].

Resta agora proceder à avaliação das hipóteses existentes para exportação e recolha de fluxos IPFIX. A solução encontrada para esta dissertação é o resultado da conjugação de duas ferramentas diferentes: *nProbe* para a exportação de fluxos e *libipfix* [104] para a sua recolha. A decisão para este sistema IPFIX foi feita com base em factores distintos para cada um dos módulos: no caso do exportador, a escolha recaiu sobre a *nProbe* por se tratar de uma aplicação bastante completa (possibilidade de recolha de uma variedade de métricas enorme (ver Apêndice F), factor que a distingue das alternativas), de uma utilização simples (como se pode ver na secção 4.4.3.3) e um suporte bastante bom da parte do seu autor. Em relação ao colector, *libipfix* revelou-se a única escolha que corresponde aos requisitos deste trabalho: o conjunto de bibliotecas de recepção e descodificação de fluxos IPFIX torna possível a implementação de um colector IPFIX na OML.

4.4.3.1 OpenIMP

Este software foi desenvolvido para fazer medidas de análise da qualidade de serviço, tais como volume de tráfego, atraso, *jitter* e pacotes perdidos. O *OpenIMP* possibilita tanto medidas activas como passivas [84], com funções de análise e de visualização [105].

O *OpenIMP* utiliza uma unidade de controlo central (controlador) e várias unidades de medida (*probes*). As *probes* são distribuídas ao longo da rede, podendo monitorizar a rede de forma passiva ou tirando medidas de desempenho activamente. As medidas a realizar são configuradas no controlador, que as distribui para as *probes*, que por sua vez enviarão de volta para o controlador (colector), através de IPFIX, os resultados medidos, sendo depois armazenados numa base de dados. Esta arquitectura pode ser observada na Figura 21 [105].

Fazendo uma breve análise às características do *OpenIMP* constata-se que pode ser aplicado em Ipv4, Ipv6 e ambientes heterogéneos, com várias funções: captura de pacotes, volume (contagem de pacotes e bytes), atraso, *jitter*, medidas de perdas em fluxos RTP de forma não intrusiva e medições IP activas de QoS (atraso e perdas tanto num só sentido como em ambos).

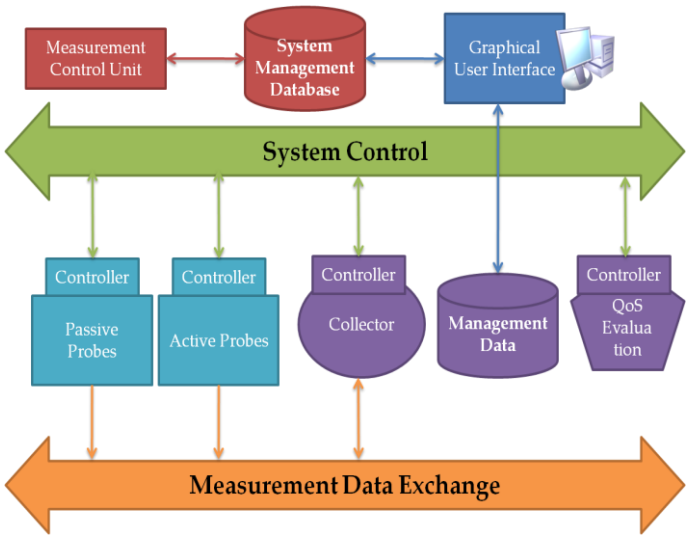

Figura 21 – OpenIMP

O sistema OpenIMP (Figura 22 [105]) é constituído pelos seguintes componentes: *probes* passivas (incapazes de gerar tráfego), filtram o tráfego recebido, fazendo operações de classificação de fluxos IP ou contagem de pacotes; *probes* activas (capazes de gerar e receber tráfego), tornam possível lidar com protocolos de camada superior como TCP ou HTTP, sendo usadas para medição de características específicas da rede, tornando ainda possível a repetibilidade da experiência; o colector, como o próprio nome indica, recolhe os dados enviados pelas *probes*; o servidor de avaliação faz o estudo dos fluxos recebidos, obtendo métricas de QoS IP, tais como *jitter* e *packet loss*; por último, a unidade de controlo de medida tem como função gerir todo o sistema de medida, por exemplo enviando as tarefas de medida para as *probes*.

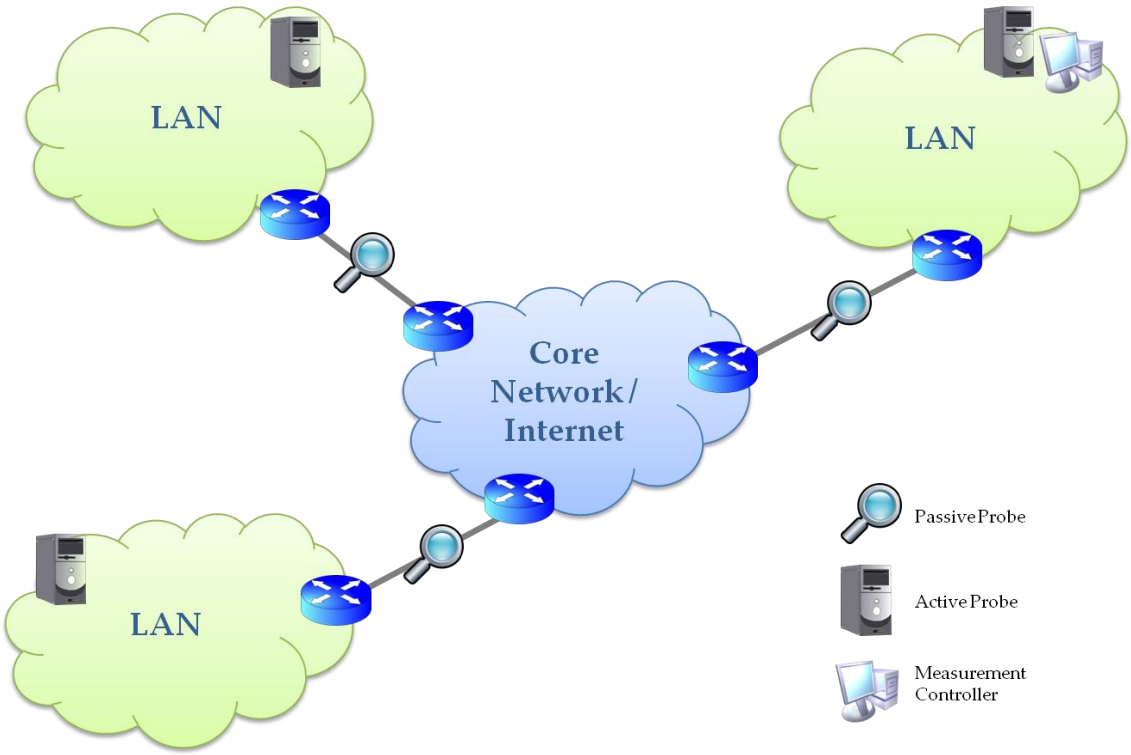

Figura 22 - Exemplo de configuração OpenIMP

Como se pode constatar na Figura 22 o *OpenIMP* possui um esquema de actuação que vai ao encontro dos pretendidos nesta dissertação. Todavia, a sua implementação revelou-se um processo complicado, não tornando possível a sua integração na OML.

4.4.3.2 libIPFIX

Como referido na secção anterior, apesar de o *OpenIMP* não permitir a implementação pretendida, possuía características proveitosas para este trabalho.

Tendo isso em conta, foi testada a *libIPFIX* (por ser uma biblioteca em C [106], com funções de exportação e colecção de fluxos IPFIX), que serviu de base à ferramenta *OpenIMP*.

Por um lado, ao analisar o colector chegou-se à conclusão que este correspondia às exigências deste trabalho. A existência de funções em C, para recepção, descodificação e armazenamento dos fluxos IPFIX torna possível a sua integração na biblioteca de medidas OML.

Por outro lado, o exportador fornecido pela *libIPFIX* é bastante limitado em relação às métricas capturadas, como se pode confirmar no modelo de resultados exposto na Tabela 4.

Exportador *libipfix*
Template ID
NFields
flowStartMilliseconds
flowEndMilliseconds
sourceIPv4Address
destinationIPv4Address
sourceTransportPort
destinationTransportPort
protocolIdentifier
ipClassOfService
packetDeltaCount
octetDeltaCount

Tabela 4 - *Template* do exportador da *libipfix*

De facto, como se pode verificar, os dados recolhidos são insuficientes, pelo que foi necessário recorrer a outra *probe*, que tratasse uma maior quantidade de informação.

4.4.3.3 nProbe

nProbe é uma *probe* de Netflow e IPFIX, para IPv4 e IPv6, caracterizada pela portabilidade para ambientes UNIX e Windows, com baixos consumos de memória e CPU, indicada portanto para ambientes com recursos limitados.

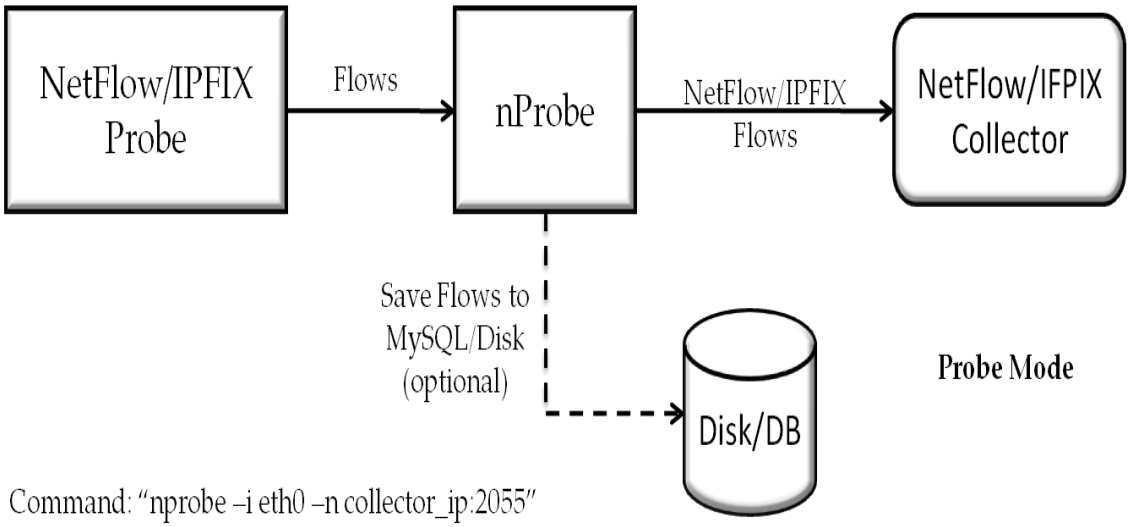

Figura 23 – nProbe

A Figura 23 [107] mostra o funcionamento da *nProbe*, que corresponde exactamente ao pretendido. A *nProbe* oferece ainda uma grande diversidade de métricas (consultar Apêndice F),

entre as quais dados relacionados com RTP [108], SIP [109], BGP [110], HTTP [111], suplantando as oferecidas pelo processo actual de recolha de resultados e tornando o sistema a implementar uma alternativa extremamente viável à solução oferecida pela OML.

A Tabela 5 é o exemplo do tipo de instrução utilizada para dar início à *nProbe*:

```
nprobe –V 10 –T –i ath1 –n 10.110.1.150:4739 "%IPV4_SRC_ADDR %IPV4_DST_ADDR
%IPV4_NEXT_HOP %INPUT_SNMP %OUTPUT_SNMP %IN_PKTS %IN_BYTES
%FIRST_SWITCHED %LAST_SWITCHED %L4_SRC_PORT %L4_DST_PORT %TCP_FLAGS
%PROTOCOL %SRC_TOS %SRC_AS %DST_AS %SRC_MASK %DST_MASK"
```

Tabela 5 - Instrução *nProbe*

Com este comando a *nProbe* dá início à recolha da informação pedida na interface ath1, através da definição do *template* que inclui dados como os endereços IP de origem e destino ou o número de bytes recebidos, exportando-a por IPFIX para o endereço 10.110.1.150, porta 4739.

Capítulo 5 – Solução Proposta

O objectivo desta dissertação é a criação de uma plataforma de monitorização de métricas na rede de testes, à qual outros utilizadores possam recorrer. Para isso foi pensada uma solução que fizesse uso do esquema e recursos disponíveis pela OML, estendendo-a com a integração da exportação de fluxos IPFIX.

Após as análises conduzidas ao longo dos Capítulos 3 e 4 foi possível desenhar o esquema da solução a implementar, exposto na Figura 24.

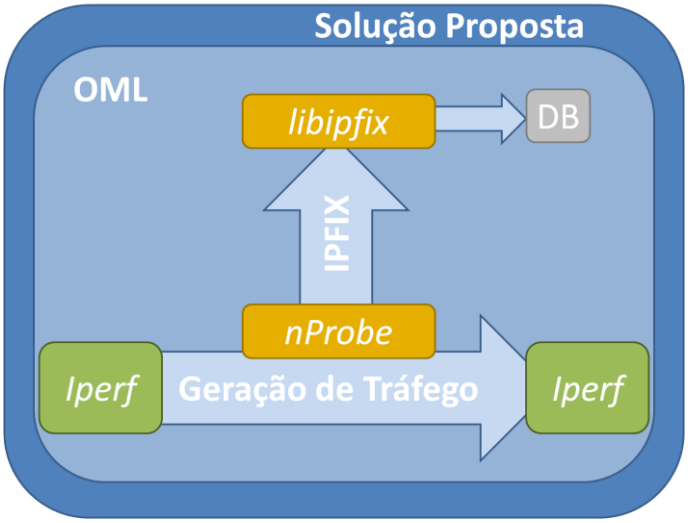

Figura 24 - Solução Proposta

Com a solução desenhada deixa de ser necessário aos utilizadores da biblioteca OML instrumentarem as suas aplicações, sendo apenas necessária a configuração do exportador, que como verificado na secção 4.4.3.3, é uma tarefa simples.

O sistema proposto consiste então de dois módulos principais, o colector e o exportador. Nas duas secções seguintes serão abordadas as implementações para cada um dos casos.

5.1 Módulo 1 - Exportador

Para o módulo do exportador a escolha recai sobre a *nProbe*: foi instalada nos nós esta ferramenta, capaz de exportar fluxos IPFIX para o colector definido. Para a sua integração no processo de execução de experiências na OMF foram tomados os seguintes passos (consultar Apêndice E):

1 – Definição da Aplicação

Neste primeiro passo é criada a AD, chamada de *wrapper* à volta da *nProbe*. Este código, que contem elementos como o directório da aplicação e as propriedades disponíveis ao utilizador para configuração (endereço de destino dos fluxos, i.e. o endereço IP do colector, a versão dos fluxos enviados, a interface de onde são capturados os pacotes, o *template* onde são definidas as métricas a registar e o nível de verboso), é gravado no repositório em */test/app* com o nome *nprobe.rb*.

2 – Definição do Protótipo

De seguida é criado o PD, que faz uso do "invólucro" criado anteriormente. É neste script que é feita a associação entre a aplicação instalada e o protótipo criado, sendo ainda possível definir alguns valores padrão como foi o caso para o endereço do colector (que será um servidor fixo), o nível de verboso, e o *template* usado, 10, que corresponde a IPFIX. A PD é guardada em */test/proto*, com o nome de *ipfix_nprobe.rb*.

3 – Definição da Experiência

Por fim, aquando a criação da ED, é utilizado o protótipo criado no passo anterior. Na ED contida no apêndice referido pode-se confirmar a inserção da ferramenta *nProbe*, devidamente preparada nos passos anteriores, num exemplo de uma experiência que utiliza a ferramenta *Iperf* para geração de tráfego.

Concluído este processo de integração, a ferramenta *nProbe* encontra-se ao dispor de qualquer utilizador, mediante a inclusão na ED das configurações necessárias.

5.2 Módulo 2 - Colector

O segundo módulo é o colector IPFIX: é neste módulo que reside a integração na OML.

O funcionamento actual da OML consiste na abertura de um *socket* que aguarda a recepção da informação, em XML [112], das medidas tiradas (Figura 25). Tirando proveito desse esquema, a solução desenhada resume-se à adição de um novo manipulador de cliente, que abra um novo *socket* e aguarde os fluxos IPFIX, ou seja, a criação de uma nova função manipulador de cliente com as funções de colector da *libIPFIX*, como mostra a Figura 26.

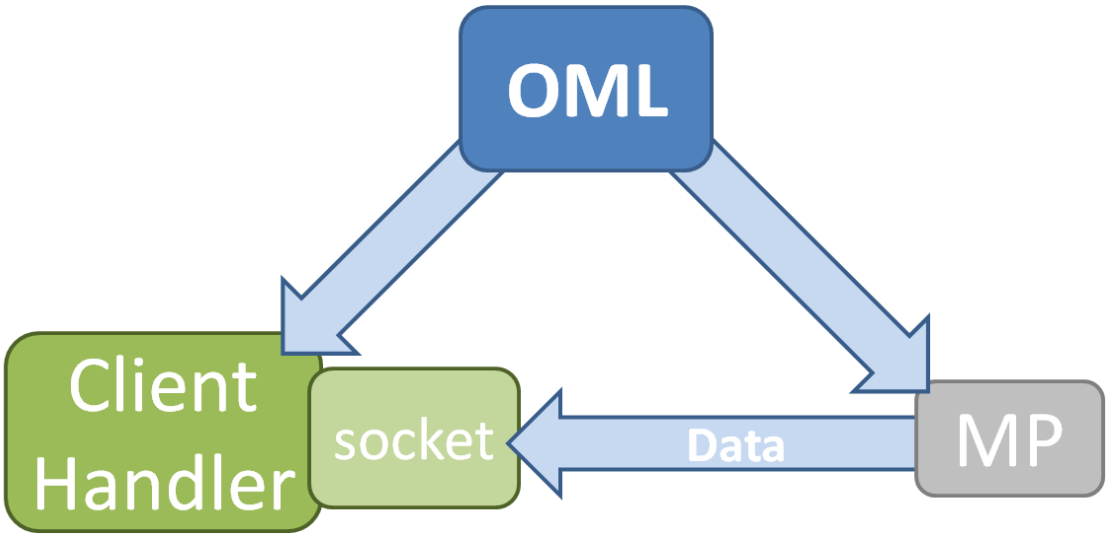

Figura 25 - Solução OML actual

O novo manipulador de cliente, *ipfix_client_handler*, é criado paralelamente ao *client_handler* já existente na biblioteca OML (ver Figura 25). Desta forma é possível atingir uma integração harmoniosa com a arquitectura OML que fica salvaguardada de quaisquer modificações ao seu manipulador de cliente.

O *ipfix_client_handler* desenvolvido faz o processamento dos fluxos IPFIX recebidos começando por criar uma base de dados denominada "IPFIX" no directório /tmp/ do colector, dando depois início ao processo de descodificação dos pacotes recebidos, que consiste na recepção dos fluxos IPFIX, criação das tabelas apropriadas na base de dados *sqlite* e seu preenchimento com as métricas enviadas pela *probe*.

A solução encontrada para a integração do manipulador de cliente criado, que permitisse a sua utilização sem causar quaisquer interferências à execução normal da OML, foi a sua inclusão como uma *thread* no código principal do servidor. Assim, é possível a execução concorrente de dois processos: o *client_handler* e o *ipfix_cliente_handler*, dotando o servidor OML da capacidade simultânea de receber resultados enviados pelos pontos de medida OML e fluxos IPFIX, contendo os dados enviados pelos exportadores.

A Figura 26 é uma representação gráfica da introdução de alterações no código do servidor OML, permitindo a recepção de fluxos IPFIX e sua descodificação.

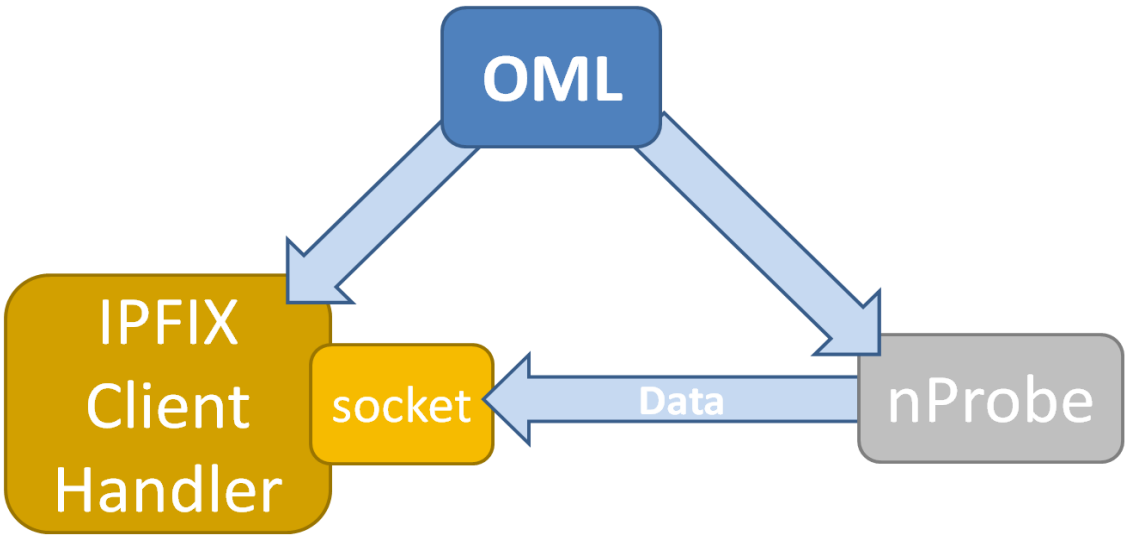

Figura 26 - Implementação da solução

A *thread* implementada, *ipfix_client_handler*, permite a descodificação dos fluxos IPFIX recorrendo a funções com base nas bibliotecas da *libipfix*:

- *ipfix_sqlite_newsource*
  - Criação da tabela *ipfix_exporters* e seu preenchimento, com valores como o endereço IP do exportador de fluxos.
- *ipfix_sqlite_newmsg*
  - Geração da tabela *ipfix_messages* e seu preenchimento com valores de identificação da experiência e do exportador.
- *ipfix_sqlite_trecord*
  - Formação da tabela *ipfix_templates* que possui o *template* das mensagens recebidas, e usa-o para a criação da tabela usada pela próxima função.
- *ipfix_sqlite_drecord*
  - Tem a seu cargo a descodificação dos fluxos e armazenamento dos resultados na tabela criada na função anterior (que terá um nome variável, como por exemplo ipfix_1_2_8_c_15_16_17_22_28_v8b30_9a_9b_9c_9d) para além da criação da tabela *ipfix_messages*, com campos de identificação.

Após a implementação acima descrita obtém-se um servidor OML com suporte para IPFIX, ou seja, ao correr o servidor OML passa-se a ter não só um servidor pronto a receber as métricas enviadas pelos pontos de medida OML mas também uma extensão IPFIX, para recepção de fluxos enviados por um exportador.

5.3 Solução Final

Após a definição da arquitectura proposta e sua implementação obtém-se a solução final da Figura 27.

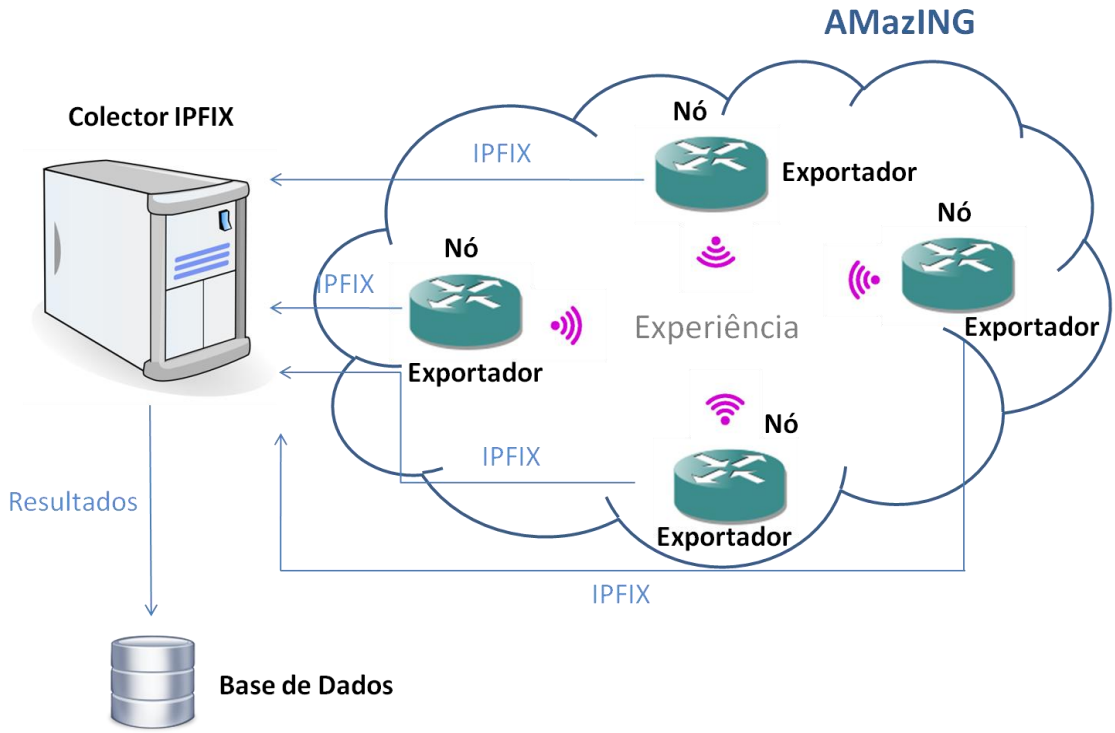

Figura 27 - Solução Final

Como se pode ver pela Figura 27, em cada nó da rede AMazING está instalado o exportador, *nProbe* (ver secção 5.1), enquanto no servidor fora da rede de testes se encontra implementado o colector (conforme descrito na secção 5.2). No decorrer das experiências na rede os exportadores irão capturar as métricas definidas no template, enviando os fluxos IPFIX por Ethernet para o servidor, que recebe os pacotes IPFIX, descodificando-os e armazenado os resultados numa base de dados para posterior consulta.

Em suma, após a implementação da solução proposta alcançou-se uma plataforma de medidas capaz de capturar métricas remotamente, facultando uma visão sobre o desempenho da rede experimental a todos os utilizadores.

No capítulo seguinte será analisada a viabilidade desta solução, comparando os resultados obtidos com os resultados adquiridos através da biblioteca de medidas OML.

Capítulo 6 - Resultados

Este capítulo contém o trabalho experimental desenvolvido com a solução descrita no Capítulo 5. Ao apresentar a solução proposta nesta dissertação como uma alternativa viável ao método presente na OML para recolha de métricas é fundamental a demonstração, através da experimentação, de que os resultados produzidos são equiparáveis aos obtidos através da OML.

As métricas avaliadas ao longo deste capítulo são a taxa de transferência, taxa de pacotes perdidos e *jitter*, por serem as métricas possíveis de extrair através da plataforma OML, recorrendo ao *Iperf*, permitindo uma comparação entre os resultados adquiridos através dos dois métodos. O capítulo divide-se assim em quatro secções: numa secção, são descritas as condições em que os testes foram realizados; outra secção é explicativa das experiências executadas; uma secção dedicada a cada métrica; por fim, uma secção para a comparação de recursos do sistema consumidos com a solução desenhada.

6.1 Detalhes da Realização de Testes

As experiências descritas nas secções seguintes foram realizadas na rede de testes AMazING, entre os nós 1 e 3, à frequência de 2,432 GHz, sob condições atmosféricas favoráveis.

A nível de software utilizou-se a plataforma de gestão e controlo OMF, versão 5.2, e a biblioteca de medidas OML, versão 2.4, estendida com a implementação da solução proposta neste trabalho.

A filtragem e análise dos resultados são feitas através do método recomendado na documentação da OML [113] [114], e os resultados serão apresentados na forma de gráficos.

6.1.1 Experiência 1 - OML

Como referido previamente na secção 4.3, a ferramenta utilizada para a geração de tráfego foi o *Iperf*, por se encontrar integrada na biblioteca OML.

Na Tabela 6, estão as instruções utilizadas para executar a geração de tráfego através do *Iperf* e a recolha de resultados através da OML.

Nó 1:~# oml2-iperf -c 192.168.0.3 -u -i 0.5 -t 600 -b 100000000 --oml-id 1 --oml-exp-id testfinal --oml-server 10.110.1.150:5000 --oml-log-file finallogfile.log --oml-log-level 4 **Nó 3:~#** oml2-iperf -s -u -i 0.5 --oml-id 1 --oml-exp-id testfinal --oml-server 10.110.1.150:5000 --oml-log-file finallogfile.log --oml-log-level 4 **Servidor#** ./oml2-server -l 5000 --logfile omllogfile.log --data-dir=/tmp/

Tabela 6 - Experiência 1 - OML

O nó 1 actua como o cliente *Iperf*, gerando tráfego UDP, a uma velocidade de 100Mbit/s, enviando resultados a cada 0,5 segundos, durante 10 minutos. Os resultados serão associados ao ID "testfinal".

O nó 3 desempenha o papel de servidor *Iperf*, recebendo o tráfego UDP enviado pelo cliente, gerando relatórios de resultados a cada 0,5 segundos e enviando-os também para o servidor OML com o mesmo ID do nó 1.

Por último, o servidor é configurado para aguardar a recepção de resultados enviados para a porta 5000, guardando os resultados numa base de dados denominada *testfinal.sq3*, no directório */tmp/*. A base de dados tem várias tabelas, sendo a relevante para a análise das métricas pretendidas a tabela *iperf_UDP_Rich_Info* [115], que corresponde aos resultados enviados pelo servidor *Iperf*, que possui os campos dispostos na Tabela 7.

iperf_UDP_Rich_Info	
oml_sender_id	ID do nó que envia os resultados
oml_seq	Número da medida efectuada
oml_ts_client	Marca de tempo adicionada pelo cliente OML
oml_ts_server	Marca de tempo adicionada pelo servidor OML
Begin_interval	Início do intervalo da medida (segundos)
End_interval	Fim do intervalo da medida (segundos)
Transfer	Dados transmitidos (bytes)
Bandwidth	Taxa de Transferência (bytes/segundo)
Jitter	Jitter (segundos)
Packet_Lost	Pacotes Perdidos
Total_Packet	Total de Pacotes
PLR	Taxa de Pacotes Perdidos (%)

Tabela 7 - *iperf_UDP_Rich_Info*

6.1.2 Experiência 2 – Solução Proposta

A experimentação da solução proposta possui uma base comum com a experiência descrita na secção 6.1.1: o servidor OML. A implementação do colector IPFIX (secção 5.2) não interfere com o funcionamento normal da OML e por tal, a sua execução é a contida na Tabela 8: o servidor OML para além de aguardar a recepção dos resultados OML também conta agora com a porta 4739 para a recepção de fluxos IPFIX.

Esta experiência diverge da anterior pela inclusão de um exportador IPFIX no nó servidor de *Iperf* (conforme a Tabela 8): o exportador escolhido no Capítulo 4, *nProbe*, será executado no nó 3, (-i) tendo como parâmetros a *interface* de onde são capturados os pacotes, (-V) a versão do fluxos a exportar (10 corresponde a IPFIX), (-T) o *template* onde é especificado o modelo dos fluxos a enviar (confira-se a Tabela 8 para uma descrição mais pormenorizada do modelo utilizado), (-n) o endereço IP e porta do servidor, (-t) tempo de vida de 1 segundo para os fluxos, (-s) o envio de fluxos expirados a cada segundo, e (-d) 1 segundo de vida para fluxos inactivos.

Uma particularidade desta experiência é o facto da *nProbe* capturar as métricas *jitter* e taxa de pacotes perdidos para tráfego RTP (conforme apresentado em [37]). Para isso foi utilizada a aplicação VLC [116] (Tabela 8) para a geração de tráfego RTP entre os nós 1 e 3. Em relação à taxa de transferência esta é obtida indirectamente através da quantidade de dados recebida num intervalo de tempo.

Nó 3:~# nprobe -i ath1 -b 2 -V 10 -T "%IPV4_SRC_ADDR %IPV4_DST_ADDR %IN_BYTES %FIRST_SWITCHED %LAST_SWITCHED %PROTOCOL %RTP_IN_JITTER %RTP_OUT_JITTER %RTP_IN_PKT_LOST %RTP_OUT_PKT_LOST" -n 10.110.1.150:4739 -t 1 -s 1 -d 1 -f "rtp port 1234"
Nó 1:~# vlc input_stream --sout '#rtp{dst=192.168.0.3,port=1234}'
Nó 3:~# vlc rtp://@192.168.0.3:1234

Tabela 8 - Experiência 2 - Solução Proposta

A base de dados com os resultados exportados pela *nProbe* é guardada no mesmo directório da experiência anterior, */tmp/*, com o nome de IPFIX. A base de dados é povoada com várias tabelas, sendo as métricas armazenadas numa tabela denominada consoante a *template* do fluxo, cujo esquema corresponde aos campos definidos no modelo da Tabela 9.

nProbe	
IPV4_SRC_ADDR	Endereço IP de origem do fluxo
IPV4_DST_ADDR	Endereço IP de destino do fluxo
IN_BYTES	Dados recebidos nos fluxos (bytes)
FIRST_SWITCHED	Tempo de início do fluxo (milissegundos)
LAST_SWITCHED	Tempo de fim do fluxo (milissegundos)
PROTOCOL	Protocolo IP
RTP_IN_JITTER	Jitter (segundos)
RTP_OUT_JITTER	Jitter (segundos)
RTP_IN_PKT_LOST	Pacotes Perdidos
RTP_OUT_PKT_LOST	Pacotes Perdidos

Tabela 9 - Resultados *nProbe*

6.2 Taxa de Transferência

A taxa de transferência é uma métrica de grande importância: o seu valor corresponde à quantidade de dados que a rede de testes suporta, por unidade de tempo. A Figura 28 e Figura 29 correspondem ao gráfico da taxa de transferência da Experiência 1 e da Experiência 2, respectivamente.

Após uma primeira observação desses gráficos é possível retirar desde logo algumas ilações:

- Os valores médios obtidos são semelhantes entre as experiências;
- A dispersão dos valores obtidos também é idêntica;
- Em ambas as experiências existem alguns picos de descida da taxa de transferência.

Fazendo uma análise estatística mais rigorosa obtiveram-se os valores contidos na Tabela 10.

Taxa de Transferência – Tratamento Estatístico	Experiência 1	Experiência 2
Média (Mbit/s)	27,70	27,25
Mediana (Mbit/s)	28,22	27,40
Moda (Mbit/s)	28,41	28,96
Desvio Padrão (Mbit/s)	1,29	1,44
Variância ([Mbit/s]²)	1,66	2,08
Coeficiente de Variação (%)	4,65	5,30
Máximo (Mbit/s)	29,40	29,23
Mínimo (Mbit/s)	20,20	21,04

Tabela 10 - Taxa de Transferência - Dados Estatísticos

Como se pode constatar ao analisar a Tabela 10 os resultados obtidos através da solução proposta (Experiência 2) são equiparáveis aos adquiridos pela OML (Experiência 1), bastante próximos de um valor médio de 27,5 Mbit/s.

O desvio padrão registado em ambas as experiências, em conjunto com o estudo dos gráficos da Figura 28 e da Figura 29, indica uma dispersão considerável, o que não é imprevisto: apesar da localização da rede de testes AMazING a isolação não é completa, e a operar à frequência de 2,432 GHz existem várias fontes de interferência (interferências externas, como por exemplo outras antenas próximas) que explicam a diferença de cerca de 7 Mbit/s entre os valores médios e o

valor mínimo da taxa de transferência (picos referidos na análise preliminar), e uma consequente dispersão de resultados. Todavia, o coeficiente de variação, para as duas experiências, possui um valor reduzido (aproximadamente 5%) o que significa que os resultados não possuem uma dispersão demasiado elevada em relação à sua média.

Aprofundando ainda mais a avaliação dos resultados obtidos foram desenhados os histogramas das experiências realizadas, que se podem visualizar na Figura 30 e Figura 31.

Na Figura 30 está exposto o histograma da frequência absoluta. Apesar do número de amostras das experiências ser diferente, devido ao facto de na Experiência 1 serem recolhidos dois resultados por segundo e na Experiência 2 ser um resultado por segundo, é possível concluir que os resultados se encontram centrados nos valores 28,5 e 29 Mbit/s.

Na Figura 31 está disposto o histograma da frequência das amostras em relação ao total de amostras, o que proporciona uma melhor comparação entre as duas experiências, onde se confirma a disposição dos resultados em torno dos 28,5 Mbit/s para a Experiência 1 e dos 29 Mbit/s para a Experiência 2, consistente com os valores calculados para a moda.

A distribuição observada em ambos os casos pode ser descrita como uma distribuição de Rice [117], em que as componentes dominantes serão 28,5 e 29 Mbit/s para a Experiência 1 e Experiência 2, respectivamente. Esta distribuição pode ser interpretada como consequência da dispersão das ondas de transmissão entre os nós, em que o valor dominante está relacionado com o percurso directo da onda e os valores dispersos causados por reflexões.

Em suma, esta primeira métrica analisada revela-se consistente com o esperado, demonstrando uma proximidade entre os dois processos experimentais, como pretendido.

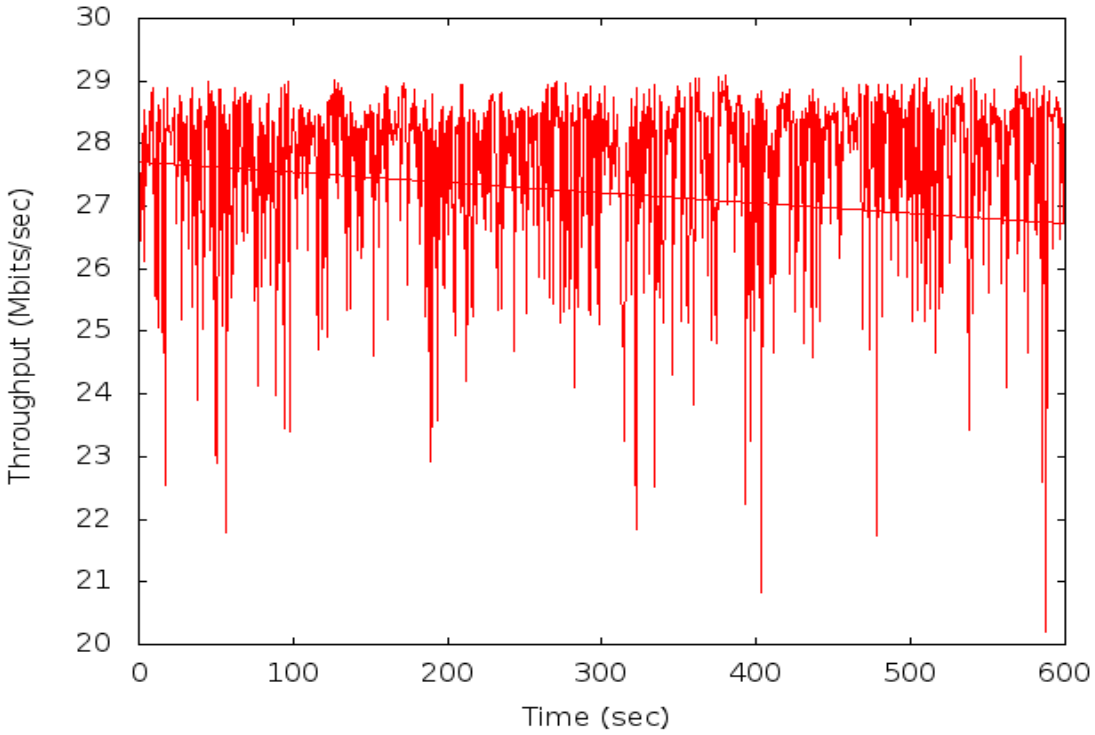

Figura 28 - Experiência 1 - Taxa de Transferência

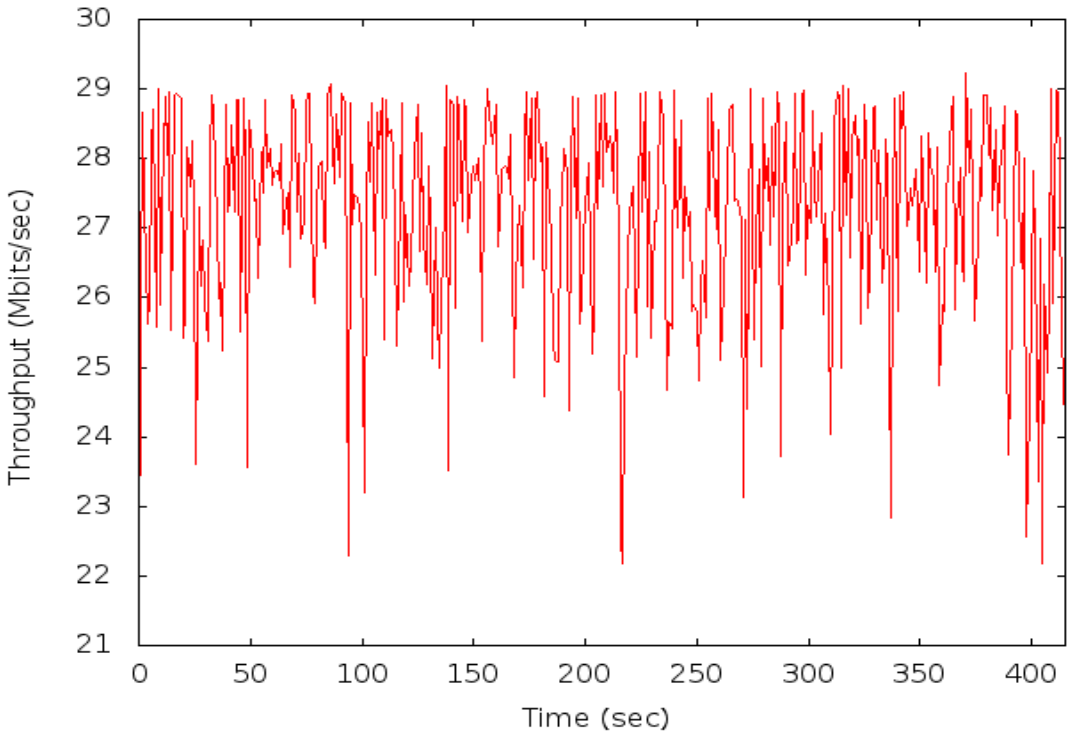

Figura 29 - Experiência 2 - Taxa de Transferência

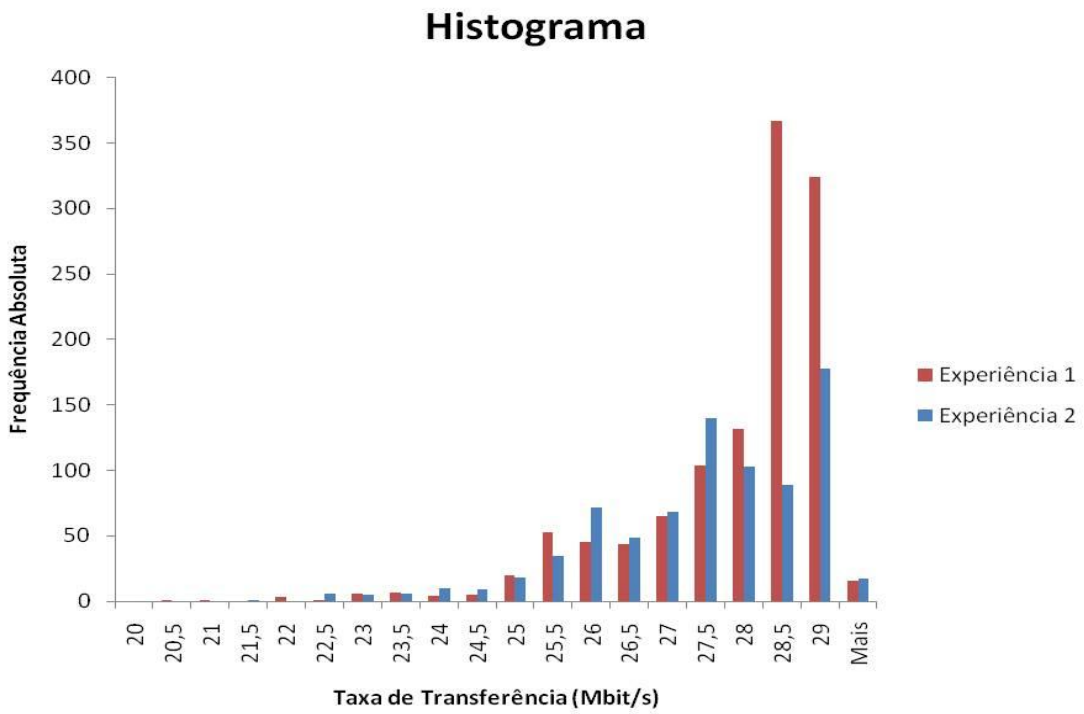

Figura 30 – Frequência Absoluta - Taxa de Transferência

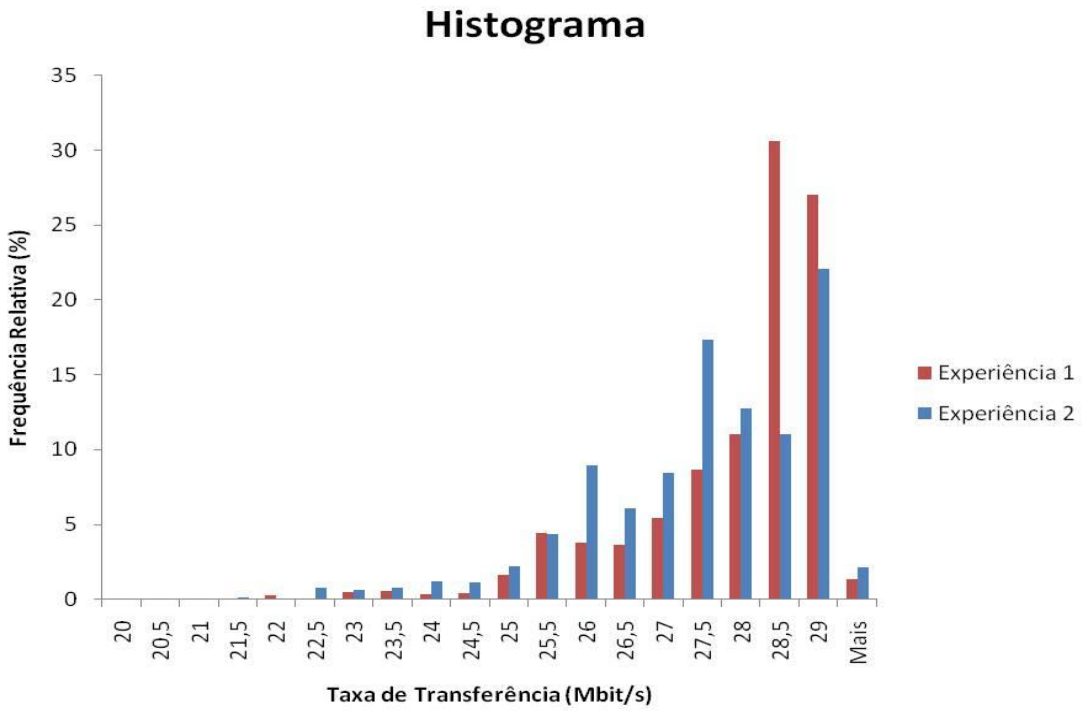

Figura 31 – Frequência Relativa - Taxa de Transferência

6.3 Taxa de Pacotes Perdidos

A segunda métrica investigada é a taxa de pacotes perdidos, também de grande relevância na avaliação do desempenho de uma rede, visto que se trata de um parâmetro que pode inviabilizar a rede. A Figura 32 e Figura 33 correspondem ao gráfico da taxa de pacotes perdidos da Experiência 1 e da Experiência 2, respectivamente.

Através de uma análise preliminar desses gráficos é possível tirar algumas conclusões:

- O aspecto mais saliente é a dispersão elevada dos resultados;
- A Experiência 1 possui picos de perdas superiores ao da Experiência 2.

Os elementos fornecidos pela Figura 32 e Figura 33 revelam-se insuficientes para o estudo da taxa de pacotes perdidos, sendo portanto necessária uma análise estatística mais detalhada, conforme o exposto na Tabela 11.

Taxa de Pacotes Perdidos – Tratamento Estatístico	Experiência 1	Experiência 2
Média (%)	2,74	4,52
Mediana (%)	0	3,88
Moda (%)	0	3,04
Desvio Padrão (%)	4,43	3,59
Variância (%²)	19,65	12,86
Coeficiente de Variação (%)	161,92	79,35
Máximo (%)	26,84	16,21
Mínimo (%)	-0,09	0

Tabela 11 - Taxa de Pacotes Perdidos - Dados Estatísticos

Os dados estatísticos da Tabela 11 vêm corroborar as observações feitas acerca dos gráficos da taxa de pacotes perdidos. Existe de facto uma dispersão bastante elevada de resultados o que pode ser derivado quer de interferências externas quer de variações de força do sinal dos nós [118]. Em consequência de dispersões destas dimensões (desvio padrão superior à média na primeira Experiência) os coeficientes de variação assumem valores bastante elevados, muito maiores que os obtidos para a taxa de transferência.

O valor da média da taxa de pacotes perdidos da solução proposta revela-se superior ao obtido através da OML. Porém não se trata de uma diferença que inviabilize a solução: apesar de

ser um aumento de cerca de 60% em relação ao registado na Experiência 1, em termos absolutos trata-se de uma percentagem inferior a 5%.

É de salientar o registo de um valor mínimo negativo na Experiência 1. Após uma análise da base de dados da Experiência 1, que contem os pacotes perdidos em valores absolutos, constata-se que um valor negativo de -0.09% de taxa de pacotes perdidos corresponde a -1 pacote perdido. Um valor negativo para a taxa de pacotes perdidos significa que num dado momento o número de pacotes recebidos é superior ao número de pacotes enviados. Tal anomalia deve-se a uma sobreposição das janelas temporais de análise de dados, causada pelo reduzido intervalo de amostragem da Experiência 1 (0.5 segundos, conforme exposto na secção 6.1.1) que em conjunto com uma variação no atraso provoca a recepção de pacotes do período de amostragem anterior na janela temporal seguinte.

A Figura 34 e Figura 35 são histogramas das medidas registadas para a taxa de pacotes perdidos. A análise destas figuras pode providenciar uma maior transparência dos dados estatísticos da Tabela 11.

Tanto através da análise da frequência absoluta, Figura 34, como da frequência relativa, Figura 35, é perceptível uma predominância do valor zero na Experiência 1, o que justifica uma moda com valor nulo e uma média inferior à da segunda Experiência, e em termos práticos se traduz numa medida em que não existe quaisquer pacotes perdidos.

Em ambas as experiências, os valores registados encontram-se maioritariamente entre 0% e 6%, havendo um declínio dos valores registados com taxa de pacotes perdidos superior a 6%, o que demonstra que com ambos os processos se observam resultados praticáveis.

Em conclusão, a taxa de pacotes perdidos medida revelou uma elevada dispersão, sendo no entanto composta por valores similares entre experiências.

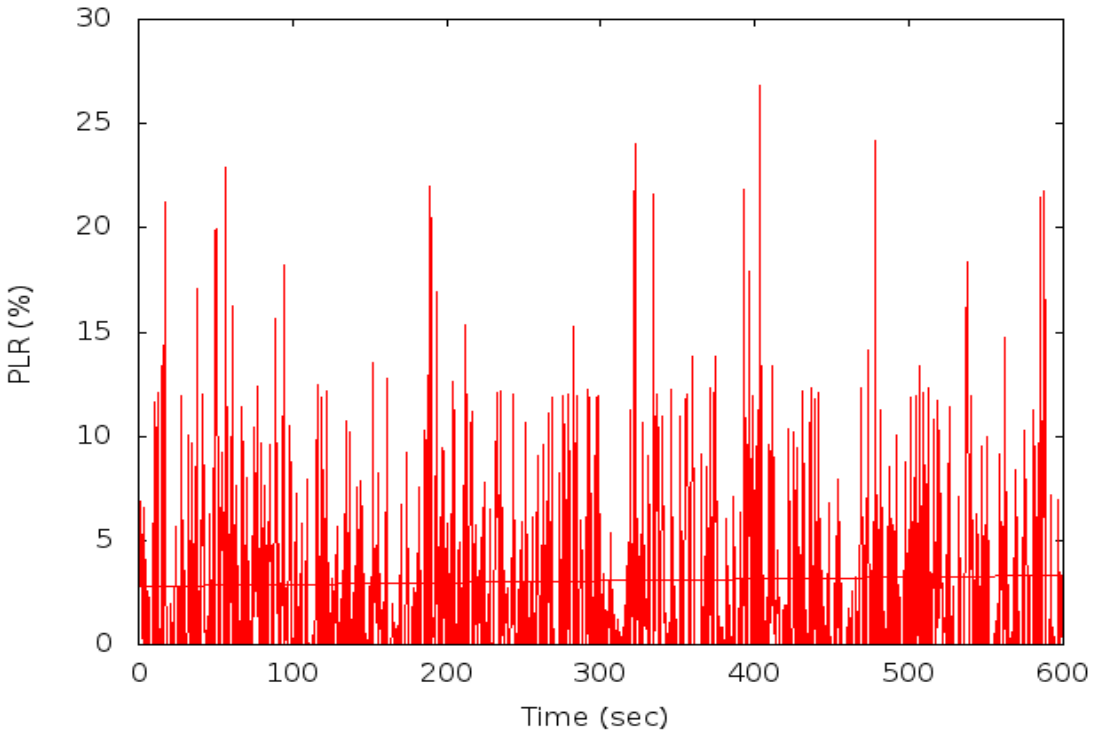

Figura 32 - Experiência 1 - Taxa de Pacotes Perdidos

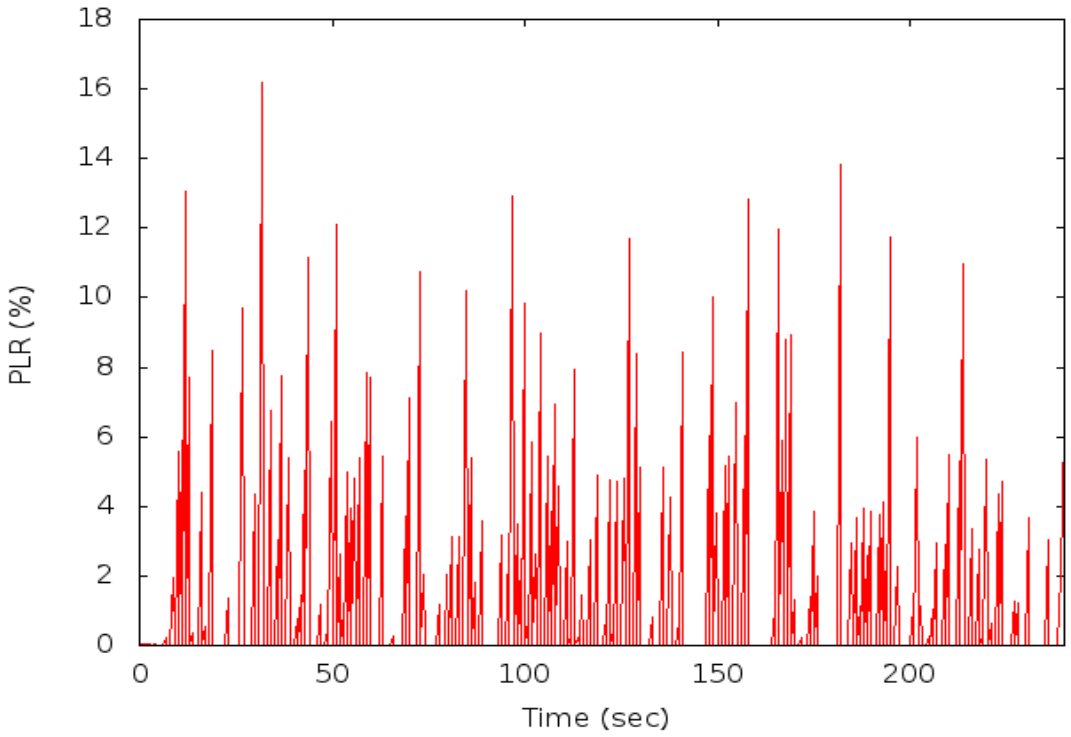

Figura 33 - Experiência 2 - Taxa de Pacotes Perdidos

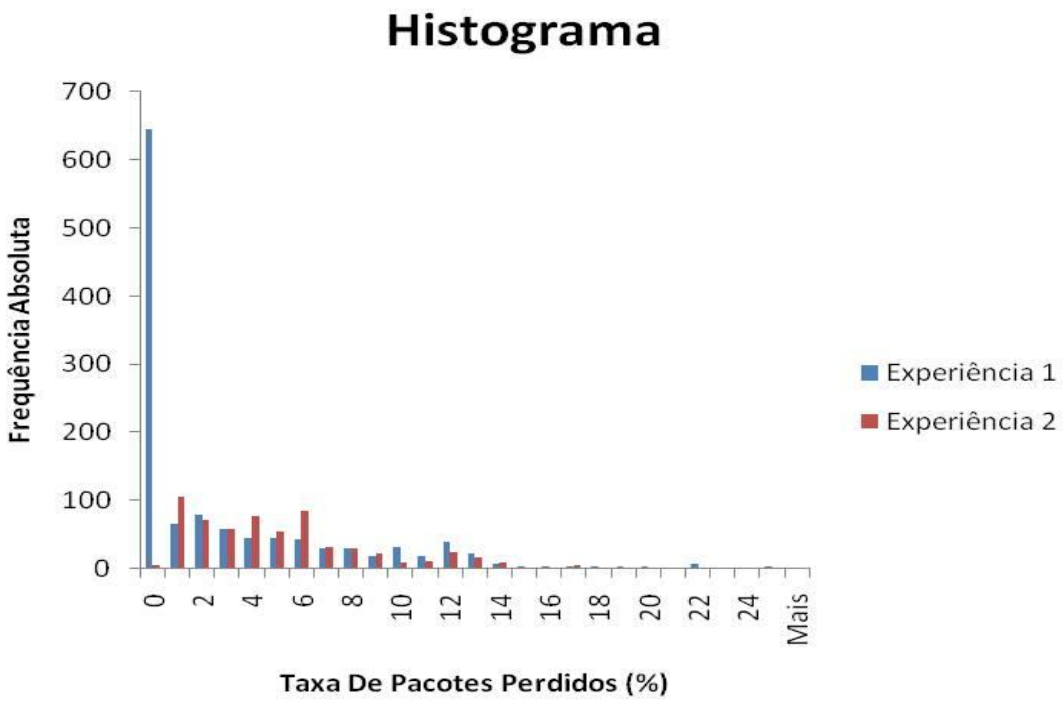

Figura 34 – Frequência Absoluta - Taxa de Pacotes Perdidos

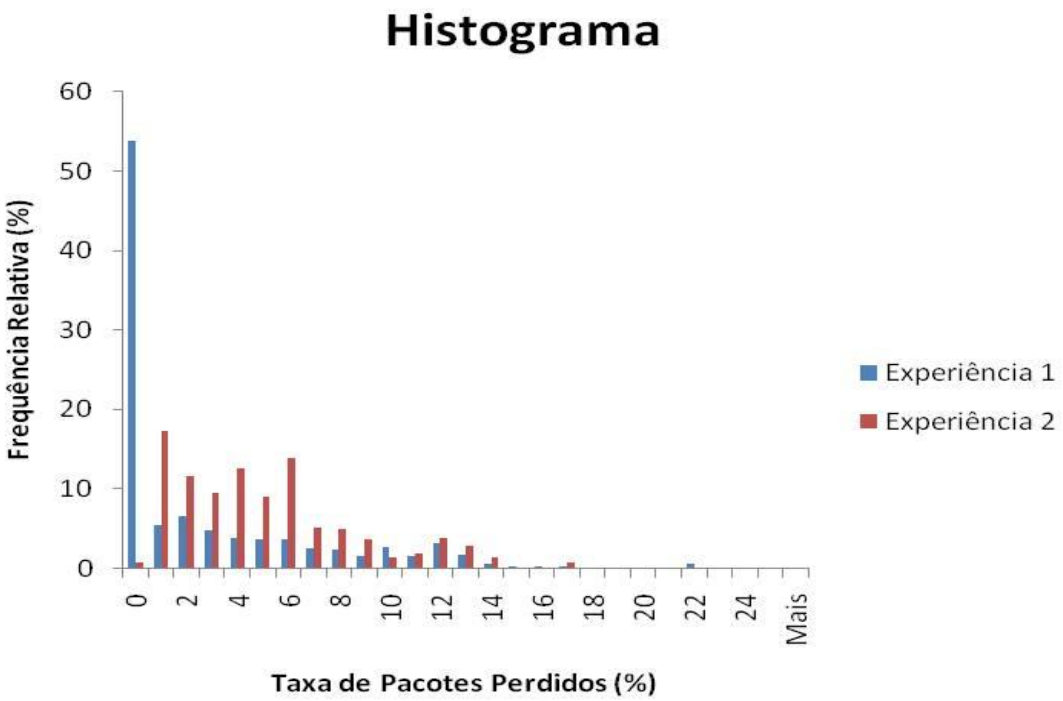

Figura 35 – Frequência Relativa - Taxa de Pacotes Perdidos

6.4 *Jitter*

Por fim, é avaliado o desempenho dos processos de medição da métrica *jitter*, que pode ser interpretado como a variação do atraso entre pacotes (ver secção 3.1.5). Na Figura 36 está representado o gráfico do *jitter* obtido durante a Experiência 1 e na Figura 37 o gráfico do *jitter* da Experiência 2.

Uma primeira análise desses gráficos permite tecer algumas considerações sobre os resultados obtidos:

- A gama de resultados é similar;
- A Experiência 2 possui uma média de valores inferior e uma dispersão superior à Experiência 1.

Verifiquemos então estas observações com um estudo estatístico dos resultados, conforme o apresentado na Tabela 12.

***Jitter* – Tratamento Estatístico**	**Experiência 1**	**Experiência 2**
Média (s)	0,75	0,52
Mediana (s)	0,67	0,05
Moda (s)	0,50	0
Desvio Padrão (s)	0,29	0,75
Variância (s^2)	0,08	0,57
Coeficiente de Variação (%)	38,25	145,0248
Máximo (s)	3,24	3,97
Mínimo (s)	0,27	0

Tabela 12- *Jitter* - Dados Estatísticos

Como inicialmente apontado verifica-se que os resultados obtidos para a solução implementada são semelhantes aos da OML, cerca de 30% inferiores devido à existência de vários resultados a zero, de acordo com a moda, conferindo aos valores obtidos com recurso ao IPFIX um nível de fiabilidade pertinente.

Confirma-se também a existência de uma dispersão mais elevada na Experiência 2, que pode novamente ser causada por interferências, como as descritas nas secções de análise da taxa de transferência e taxa de pacotes perdidos, resultando em coeficientes de variação elevados, à semelhança da Experiência 1 na secção 6.3.

É ainda possível, através da observação da Tabela 12, constatar que o processo implementado consegue atingir o valor mínimo nulo de *jitter*, sendo no entanto a Experiência 1 a alcançar o *jitter* máximo mais baixo, apesar da média superior.

Os histogramas dos resultados conseguidos para o *jitter* encontram-se representados na Figura 38 e Figura 39, para a frequência absoluta e relativa respectivamente, e permitem aprofundar a avaliação destes resultados.

Como constatado anteriormente, na Experiência 2 foi possível obter um valor nulo para o *jitter*, sendo de facto o resultado com maior incidência (moda), o que explica a média inferior da segunda Experiência.

A disposição dos histogramas é semelhante para ambas as experiências, com a diferença de que o valor com maior frequência da Experiência 1 é 0,6 segundos ao invés dos 0 segundos da outra Experiência, verificando-se depois um decréscimo gradual dos valores de *jitter* em ambos os casos, tendo uma frequência quase insignificante para resultados superiores a 1,5 segundos.

Concluindo, esta última métrica analisada confirma mais uma vez a validade dos resultados obtidos pela solução apresentada nesta dissertação, apesar de uma dispersão de resultados novamente elevada.

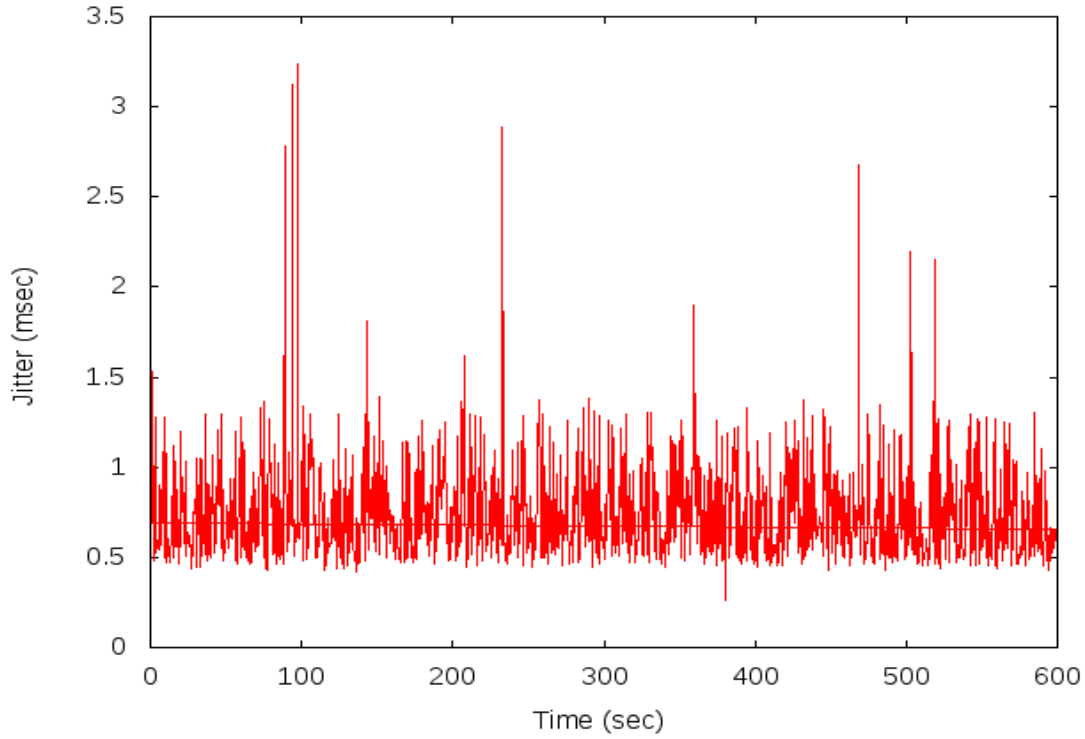

Figura 36 - Experiência 1 - *Jitter*

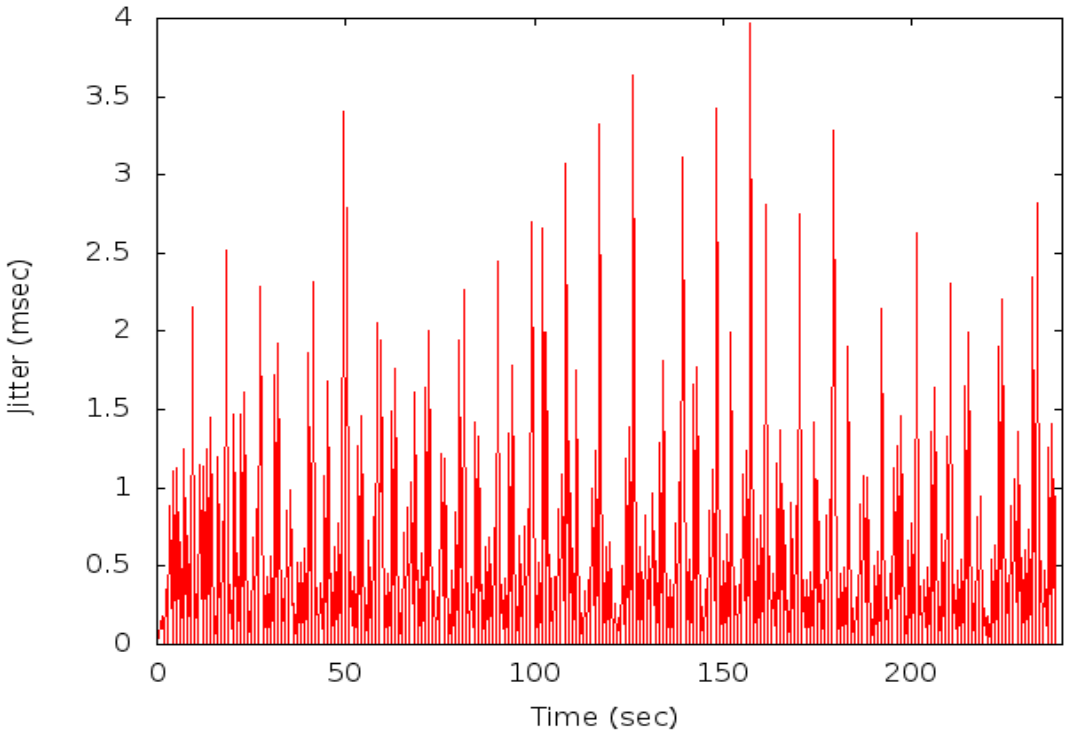

Figura 37 - Experiência 2 - *Jitter*

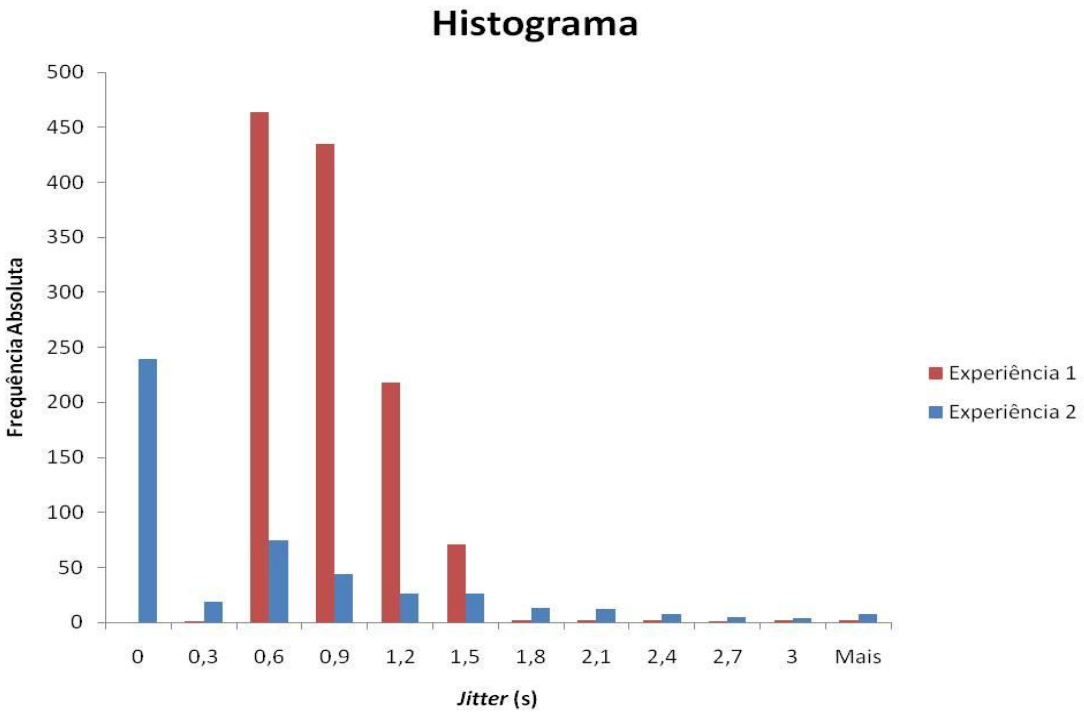

Figura 38 – Frequência Absoluta - *Jitter*

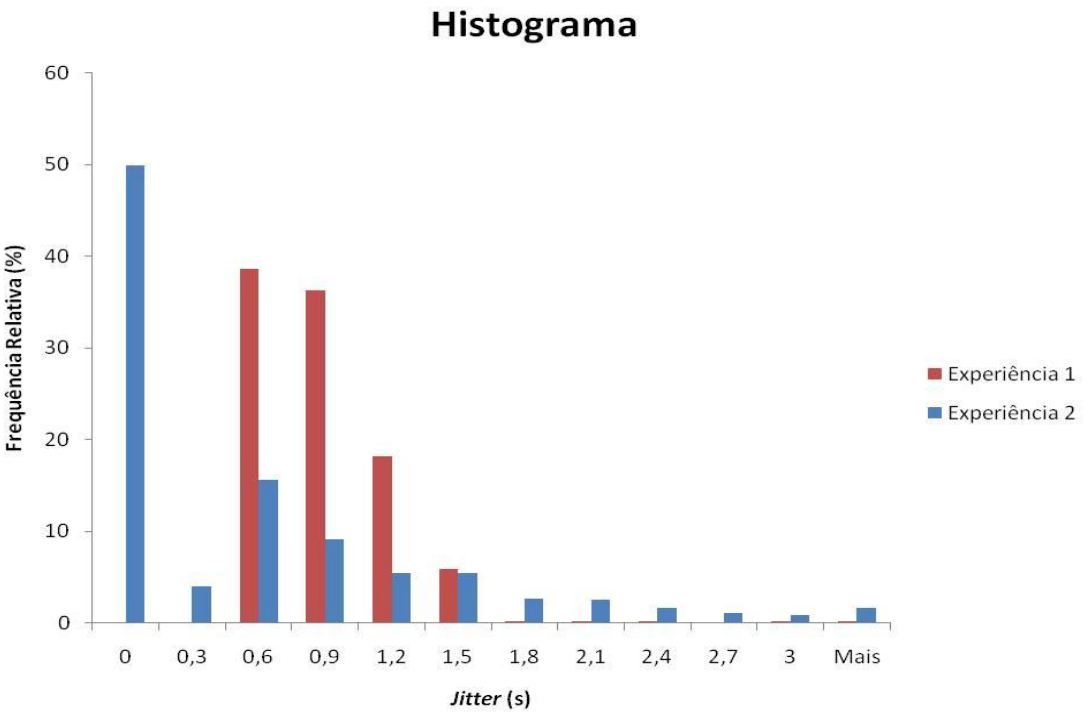

Figura 39 – Frequência Relativa - *Jitter*

6.5 Consumo de Recursos

Para além de uma análise comparativa entre os resultados obtidos através da plataforma de medidas OML original e a versão alterada com a inclusão de um colector IPFIX foi também feito um estudo do impacto no consumo de recursos do sistema que esta implementação acarreta.

A Tabela 13 possui a média dos valores registados para a percentagem de CPU ocupado e para a quantidade de memória livre durante a execução das experiências.

	CPU Ocupado (%)	**Memória Livre (Kbytes)**
Experiência 1	2,511962	244381
Experiência 2	5,513011	242923,6877

Tabela 13 - Consumo de Recursos

No caso do CPU verifica-se um aumento de cerca de 120% de ocupação que no entanto não tem quaisquer repercussões no sistema, já que trata de um valor de ocupação de cerca de 5% e, portanto, insignificante.

No segundo caso, da memória, é necessário proceder ao raciocínio inverso: passou-se de 244,381Mbytes de memória livre para, aproximadamente, 242,924Mbytes, o que representa uma ocupação cerca de 1,5Mbytes superior. Novamente, este é um valor sem qualquer expressão no desempenho total do nó.

Como se pode constatar existe de facto um aumento no consumo dos recursos do nó. Contudo, trata-se de aumentos de valor bastante reduzido e portanto desprezíveis, não anulando a validade da solução proposta.

Capítulo 7 – Conclusão

Neste trabalho foi apresentada uma solução alternativa à recolha de métricas em redes experimentais através da biblioteca de medidas OML, baseada no protocolo de transporte IPFIX. A solução desenvolvida consiste numa extensão da plataforma OML, acrescentando-lhe a capacidade de receber fluxos IPFIX e descodificá-los, armazenando os resultados numa base de dados. A exportação dos fluxos IPFIX é da responsabilidade da ferramenta *nProbe* e as funções para a implementação do colector da *libipfix*.

A validade da solução proposta foi comprovada através da sua experimentação e comparação com resultados obtidos através da plataforma OML original. O sistema desenvolvido é assim uma alternativa viável ao processo actual de recolha de medidas dada a fiabilidade dos seus resultados que, como verificado, são muito próximos dos produzidos pela biblioteca OML. Concluiu-se ainda que a inclusão desta solução não introduz efeitos nefastos significativos no consumo de recursos dos nós.

Criou-se assim uma extensão para a plataforma de medidas OML, adicionando o suporte IPFIX. Esta solução possui a vantagem de não ser necessária a alteração da aplicação a testar ou a criação de um *wrapper* para a mesma: o colector IPFIX é iniciado por defeito, sem qualquer intervenção do utilizador, com o servidor OML, a aguardar a recepção de fluxos, ao passo que para iniciar o exportador basta acrescentar na descrição da experiência as instruções adequadas. O IPFIX vem assim dotar a plataforma de medidas da rede de testes de uma maior flexibilidade e simplicidade de uso.

Existe também um outro factor de grande importância que aumenta o interesse do trabalho desenvolvido: o facto de com IPFIX, e recorrendo à *nProbe*, ser possível a aquisição de uma maior variedade de métricas, bastante superior às possíveis com a OML.

O desenvolvimento da solução apresentada nesta dissertação e a sua implementação permitiu uma aprendizagem aprofundada em diferentes áreas tecnológicas.

Com o estudo inicial das várias redes de testes existentes foi possível constatar os sistemas actualmente implementados para recolha de métricas, identificar as suas falhas e desenhar uma solução que permita colmatar essas lacunas e melhorar as plataformas existentes.

A investigação em torno das métricas de desempenho de redes e as ferramentas para as medir foram de grande importância para este trabalho, permitindo um maior conhecimento do panorama actual da avaliação de redes sem fios.

Por fim, a análise de protocolos de transporte foi fundamental visto que culminou com a selecção do protocolo IPFIX, que correspondeu às necessidades e objectivos desta dissertação.

A conjugação do conhecimento adquirido com o estudo das áreas referidas conferiu a esta dissertação a possibilidade de ser integrada numa futura distribuição oficial da plataforma OML bem como a possibilidade de servir de base a futuros artigos.

A nível pessoal esta dissertação foi um trajecto de enriquecimento quer a nível da experiência adquirida numa área proeminente como as redes de telecomunicações, quer a nível

pessoal pelo envolvimento com todos os colaboradores que contribuíram para o sucesso deste projecto, quer ainda pelo sentimento de realização atingido com o desfecho desta dissertação.

Bibliografia

[1] IEEE 802.11TM WIRELESS LOCAL AREA NETWORKS. (2010, Outubro) http://www.ieee802.org/11/. Online.

[2] Wolfgang Kiess and Martin Mauve. (2006, Janeiro) A Survey on Real-World Implementations of Mobile Ad-Hoc Networks. http://faculty.kfupm.edu.sa/coe/mayez/Selected-papers-March-7-2010/E-A%20survey%20on%20real-world%20implementations%20of%20mobile%20adhoc%20networks.pdf.

[3] The Network Simulator NS-2. (2010, Outubro) http://www.isi.edu/nsnam/ns/. Online.

[4] Abas Md Said, Halabi Hasbullah Muhammad Imran. (2010, Junho) A Survey of Simulators, Emulators and Testbeds for Wireless Sensor Networks. 2010 International Symposium in Information Technology (ITSim).

[5] Thierry Rakotoarivelo and Max Ott. (2008) Orbit Management Framework (OMF) - Presentation & Demonstration. http://mytestbed.net/attachments/download/3.

[6] OMF - Introduction - OMF Developer Portal. (2010, Outubro) http://mytestbed.net/projects/omf/wiki/Introduction. Online.

[7] OML - Measurement Framework - OML Overview - OMF Developer Portal. (2010, Outubro) http://mytestbed.net/wiki/2. Online.

[8] Ed. B. Claise. (2008, Janeiro) Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101.

[9] UMIC-Mesh.net - A hyrid testbed approach. (2010, Outubro) http://www.umic-mesh.net/testbed/. Online.

[10] UMIC-Mesh.net. (2010, Outubro) http://www.umic-mesh.net/home/. Online.

[11] MIT Roofnet. (2010, Outubro) http://pdos.csail.mit.edu/roofnet/doku.php. Online.

[12] Massachusetts Institute of Technology. (2010, Outubro) http://web.mit.edu/. Online.

[13] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. (2003, Setembro) A High-Throughput Path Metric for Multi-Hop Wireless Routing. ACM Mobicom Conference.

[14] Orbit. (2010, Outubro) http://www.orbit-lab.org/. Online.

[15] A. Tucholski et al. (2009, Agosto) D9.1 - Benchmarking methodology and metrics.

[16] Purdue University Wireless Mesh Network Testbed. (2010, Outubro) https://engineering.purdue.edu/MESH/. Online.

[17] BerlinRoofNet - SarWiki. (2010, Outubro) http://sarwiki.informatik.hu-berlin.de/BerlinRoofNet. Online.

[18] Mobility Management and Networking Laboratory - UC Santa Barbara. (2010, Outubro) http://moment.cs.ucsb.edu/.

[19] UC Santa Barbara. (2010, Outubro) http://www.ucsb.edu/. Online.

[20] Gatech - Wireless Mesh Networks. (2010, Outubro) http://www.ece.gatech.edu/research/labs/bwn/mesh/index.html. Online.

[21] Georgia Tech. (2010, Outubro) http://www.gatech.edu/. Online.

[22] Wireless Networking Research Testbed. (2010, Outubro) http://networks.cs.ucr.edu/testbed/index.htm. Online.

[23] UC Riverside. (2010, Outubro) http://www.ucr.edu/. Online.

[24] B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, D. Noveck S. Shepler. (2010, Outubro) Network File System (NFS) version 4 Protocol. RFC 3530.

[25] IEEE P802.3af DTE Power via MDI Task Force. (2010, Outubro) http://www.ieee802.org/3/af/. Online.

[26] Ultra High Speed Mobile Information and Communication. (2010, Outubro) http://www.umic.rwth-aachen.de/. Online.

[27] Alexander Zimmermann, Arnd Hannemann, and Tim Kosse. (2010) Flowgrind -- A new Performance Measurement Tool. Proceedings of the 22nd IEEE Global Communications Conference (GLOBECOM'10).

[28] University of Central Florida - Iperf. (2010, Outubro) http://www.noc.ucf.edu/Tools/Iperf/. Online.

[29] Netperf Homepage. (2010, Outubro) http://www.netperf.org/netperf/. Online.

[30] Flowgrind Architecture. (2010, Outubro) http://www.umic-mesh.net/research/flowgrind/architecture.html. Online.

[31] R. Thurlow. (2009, Maio) RPC: Remote Procedure Call Protocol Specification Version 2. RFC 5531.

[32] Winlab Home. (2010, Outubro) http://www.winlab.rutgers.edu/. Online.

[33] C. Demichelis and P. Chimento. (2002, Novembro) IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393.

[34] Ivan Seskar, Robert Siraccusa, Manpreet Singh Maximilian Ott. (2005) ORBIT Testbed Software Architecture: Supporting Experiments as a Service. First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05).

[35] Instituto de Telecomunicações. (2010, Outubro) http://www.it.pt/. Online.

[36] Relay Control Panel. (2010, Outubro) http://jimbo.av.it.pt/relay-panel/. Online.

[37] Luca Deri. (2010, Outubro) nBox, nProbe, n2disk - User's Guide. http://www.nmon.net/UsersGuide.pdf.

[38] Monitoring Tools for Mobile Networks. (2010, Outubro) http://moment.cs.ucsb.edu/projects.html. Online.

[39] Maurizio Molina, Andy Van Maele, and et al. (2006, Fevereiro) Network Metric Report. http://www.perfsonar.net/jra1-wiki/images/d/d7/MCF-1.13.doc.

[40] Jonathan Guerin, Marius Portmann, and Asad Pirzada. (2010, Outubro) Routing Metrics for Multi-Radio Wireless Mesh Networks. http://www.nicta.com.au/__data/assets/pdf_file/0020/15653/Routing_Metrics_for_Multi-Radio_Wireless_Mesh_Networks.pdf.

[41] Gustavo Neves Dias. (2006, Agosto) Multi-Radio - Link Quality Source Routing (MR-LQSR). http://www.ravel.ufrj.br/arquivosPublicacoes/cpe825_mono_GustavoDias.pdf.

[42] Hossam Hassanein and Audrey Zhou. (2001) Routing with Load Balancing in Wireless Ad hoc Networks. Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems.

[43] J. Postel. (1980, Agosto) User Datagram Protocol. RFC 768.

[44] J. Postel. (1981, Setembro) Internet Control Message Protocol. RFC 792.

[45] Richard Draves, Jitendra Padhye, and Brian Zill. (2004) Comparison of Routing Metrics for Static Multi-Hop Wireless Network. Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'04).

[46] Lu Xin. Bandwidth Metrics and Measurement Tools. http://web2.uwindsor.ca/courses/cs/aggarwal/HPGCGroup/Docs/bmetrics.ppt.

[47] Theodore S. Rappaport. (2002, Janeiro) Wireless Communications: Principles and Practice. Prentice Hall.

[48] Networks and Communicatiion Branch. (2010, Outubro) http://cs.itd.nrl.navy.mil/work/mgen/index.php. Online.

[49] Distributed Internet Traffic Generator D-ITG. (2010, Outubro) http://www.grid.unina.it/software/ITG/. Online.

[50] Bruce Lowekamp, Brian Tierney, and et al. (2002, Julho) A Hierarchy of Network Measurements for Grid Applications and Services. GWD-I (NMWG Internal Draft).

[51] G. Almes, S. Kalidindi, and M. Zekauskas. (1999, Setembro) A One-way Packet Loss Metric for IPPM. RFC 2680.

[52] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. (1998, Maio) Framework for IP Performance Metrics. RFC 2330.

[53] G. Almes, S. Kalidindi, and M. Zekauskas. (1999, Setembro) A Round-trip Delay Metric for IPPM. RFC 2681.

[54] G. Almes, S. Kalidindi, and M. Zekauskas. (1999, Setembro) A One-way Delay Metric for IPPM. RFC 2679.

[55] ISO 8601 summary by Markus Kuhn. (2010, Outubro) http://www.cl.cam.ac.uk/~mgk25/iso-time.html. Online.

[56] OML - Measurement Framework - Quick Start Tutorial. (2010, Outubro)

http://mytestbed.net/projects/oml/wiki/Quick_Start_Tutorial. Online.

[57] International Telecommunication Union. (2007, Novembro) Y.1540 - Internet protocol data communication service – IP packet transfer and availability performance parameters. http://www.catr.cn/radar/itut/201007/P020100707572808037661.doc.

[58] Standardization (ITU-T). (2010, Outubro) http://www.itu.int/ITU-T/. Online.

[59] Internet Engineering Task Force. (2010, Outubro) http://www.ietf.org/. Online.

[60] Dimitris Primpas, Kostas Stamos Christos Bouras. (2008) QoS Issues in a Large-scale IPv6 Network. http://ru6.cti.gr/ru6/publications/41271178.pdf.

[61] S. Casner, R. Frederick, V. Jacobson H. Schulzrinne. (2003, Julho) RTP: A Transport Protocol for Real-Time Applications. RFC 3550.

[62] Pedro Miguel Esposito, Miguel Elias M. Campista, Marcelo G. Rubinstein, and et al. (Novembro, 2008) Implementing the Expected Transmission Time Metric for OLSR Wireless Mesh Networks. Wireless Days, 2008. WD '08. 1st IFIP.

[63] Richard Draves, Jitendra Padhye, and Brian Zill. (2004) Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. Proceedings of the 10th annual international conference on Mobile computing and networking (Mobicom'04).

[64] Diego Passos and Célio V. N. Albuquerque. (2009) A Joint Approach to Routing Metrics and Rate Adaptation in Wireless Mesh Networks. Proceedings of the 28th IEEE international conference on Computer Communications Workshops (INFOCOM'09).

[65] Meng Fang. (2005, November) Routing in Wireless Mesh Networks. http://www.ecse.rpi.edu/Homepages/abouzeid/6962-05/monet22-routing-in-mesh-fang.pdf.

[66] OMF - An Introduction to OMF. (2010, Outubro) http://omf.mytestbed.net/projects/omf/wiki/An_Introduction_to_OMF. Online.

[67] SQLite Home Page. (2010, Outubro) http://www.sqlite.org/. Online.

[68] Matlab - The Language of Technical Computing. (2010, Outubro) http://www.mathworks.com/products/matlab/. Online.

[69] Microsoft Excel. (2010, Outubro) http://office.microsoft.com/en-gb/excel/. Online.

[70] OMF - Usage Overview. (2010, Outubro) http://omf.mytestbed.net/projects/omf/wiki/UsageOverview. Online.

[71] Linguagem de Programação Ruby. (2010, Outubro) http://www.ruby-lang.org/pt/. Online.

[72] (C)RUDE - RUDE & CRUDE. (2010, Outubro) http://rude.sourceforge.net/. Online.

[73] UDPmon Home Page. (2010, Outubro) http://www.hep.man.ac.uk/u/rich/net/index.html. Online.

[74] TfGen. (2010, Outubro) http://www.st.rim.or.jp/~yumo/pub/tfgen.html. Online.

[75] KUTE - Kernel-based Traffic Engine. (2010, Outubro) http://caia.swin.edu.au/genius/tools/kute/. Online.

[76] CITI Projects: The University of Michigan and Merit Internet2 Qbone Testbed. (2010, Outubro) http://www.citi.umich.edu/projects/qbone/generator.html. Online.

[77] TrafGen. (2010, Outubro) http://www-lor.int-evry.fr/~vincent/java/trafGen/trafGenEn.htm. Online.

[78] TG Information. (2010, Outubro) http://www.postel.org/tg/tg_desc.htm#tool. Online.

[79] Information Sciences Institute. (1981, Setembro) Transmission Control Protocol. RFC 793.

[80] Free Software Foundation. (2010, Outubro) http://www.fsf.org/. Online.

[81] MGEN User's and Reference Guide. (2010, Outubro) http://pf.itd.nrl.navy.mil/mgen/mgen.html. Online.

[82] Stefano Avallone, Alessio Botta, Alberto Dainotti, Walter de Donato, and Antonio Pescapé. (2009, Maio) D-ITG V. 2.7.0 - Beta2 Manual. http://www.grid.unina.it/software/ITG/codice/D-ITG-2.7.0-Beta2-manual.pdf.

[83] Antonio Pescapè, Donato Emma, Stefano Avallon, Alessio Botta, and Giorgio Ventre. (2010, Outubro) D-ITG, Distributed Internet Traffic Generator. http://www.grid.unina.it/software/ITG/.

[84] PerJuvhaugen. (2007, Maio) Exporting IP flows using IPFIX. http://research.iu.hio.no/theses/pdf/master2007/per.pdf.

[85] sFlow - Making the Network Visible. (2010, Outubro) http://www.sflow.org/. Online.

[86] Elisa Jasinska. (2006, Dezembro) sFlow - I can feel your traffic. Amsterdam Internet Exchange (AMS-IX).

[87] sFlow.org. (2003) Traffic Monitoring using sFlow. http://www.sflow.org/sFlowOverview.pdf.

[88] Peter Phaal and Marc Lavine. (2004, Julho) sFlow Version 5. http://www.sflow.org/sflow_version_5.txt.

[89] P. Phaal, S. Panchen, and N. McKee. (2001, Setembro) InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176.

[90] Inc Cisco Systems. (2010, Outubro) http://www.cisco.com/. Online.

[91] Cisco IOS NetFlow. (2010, Outubro) http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html. Online.

[92] S. Leinen. (2004, Outubro) Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX). RFC 3955.

[93] S. Floyd, D. Black K. Ramakrishnan. (2001, Setembro) The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168.

[94] Benoit Claise. (2003, Fevereiro) Evaluation Of NetFlow Version 9 Against IPFIX Requirements. http://www.ietf.org/proceedings/56/slides/ipfix-2/ipfix-2.ppt.

[95] NetFlow architecture using standalone probes. (2008, Março)

http://upload.wikimedia.org/wikipedia/commons/c/cf/NewNetFlowApproach.png. Online.

[96] Ed. B. Claise. (2004, Outubro) Cisco Systems NetFlow Services Export Version 9. RFC 3954.

[97] IP Flow Information Export (ipfix). (2010, Outubro) http://datatracker.ietf.org/wg/ipfix/charter/. Online.

[98] Ed. B. Claise. (2008, Janeiro) Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101.

[99] J. Quittek, T. Zseby, B. Claise, and S. Zander. (2004, Outubro) Requirements for IP Flow Information Export (IPFIX). RFC 3917.

[100] IPFIX. (2010, Outubro) https://www.ntt-review.jp/archive_html/200712/images/le1-Fig-1.gif. Online.

[101] T. Zseby, E. Boschi, N. Brownlee, and B. Claise. (2009, Março) IP Flow Information Export (IPFIX) Applicability. RFC 5472.

[102] IPFIX (IP Flow Information eXport). (2010, Outubro) http://www.netflow-analyser.co.uk/scrutinizer-netflow-sflow-analyser/flow-technologies/ipfix/ipfix.php. Online.

[103] Brian Trammell and Elisa Boschi. (2007, Outubro) From NetFlow to IPFIX the evolution of IP flow information export. http://nanog.org/meetings/nanog41/presentations/nanog41-ipfix.pdf.

[104] libIPFIX - How To. (2010, Outubro) http://meteor.fokus.fraunhofer.de/libipfix/doc/howto/index.html. Online.

[105] OpenIMP. (2010, Outubro) http://www.ip-measurement.org/index.php?option=com_content&view=article&id=8&Itemid=8. Online.

[106] D. Brady, M. Doran M. Banahan. (1991, Agosto) The C Book. Addison-Wesley Pub.

[107] nProbe - NetFlow/IPFIX Network Probe. (2010, Outubro) http://www.ntop.org/nProbe.html. Online.

[108] Collin Perkins. (2003, Junho) RTP: audio and video for the Internet. Addison-Wesley Professional.

[109] H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler J. Rosenberg. (2002, Junho) SIP: Session Initiation Protocol. RFC 3261.

[110] T. Li, S. Hares Y. Rekhter. (2006, Janeiro) A Border Gateway Protocol 4 (BGP-4). RFC 4271.

[111] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee R. Fielding. (1999, Junho) Hypertext Transfer Protocol -- HTTP/1.1. RFC 2616.

[112] Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau Tim Bray. (2008, Novembro) Extensible Markup Language (XML) 1.0 (Fifth Edition). http://www.w3.org/TR/REC-xml/.

[113] OMF - Analyzing Hello World Results. (2010, Outubro) http://omf.mytestbed.net/projects/omf/wiki/Analysing_results. Online.

[114] OMF - Basic Tutorial Stage 0. (2010, Outubro)
http://omf.mytestbed.net/projects/omf/wiki/basictutorialstage0. Online.

[115] OML - Measuremente Framework - Iperf OML2. (2010, Outubro)
http://omf.mytestbed.net/projects/oml/wiki/iperf_oml2. Online.

[116] VideoLan - VLC media player. (2010, Outubro) http://www.videolan.org/vlc/. Online.

[117] I. Stegun M. Abramowitz. (1965, Junho) Handbook of Mathematical Functions. Dover Publications.

[118] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. (2006) Understanding the Causes of Packet Delivery Success and Failure in Dense Wireless Sensor Networks. Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys '06).

Apêndices

Apêndice A – Instrumentação de Aplicação na OML

1 - Incluir a biblioteca oml2/omlc.h no código fonte:

#include <oml2/omlc.h>

2 - Chamar a função *omlc_init*:

int omlc_init (const char name, int* argc_ptr, const char** argv, o_log_fn logger)*

3 – Adicionar pontos de medida (*omlc_add_mp()*):

OmlMP mp = omlc_add_mp ("packet_info", mp_def);*

4 – Chamar *omlc_start()* para dar início ao processo de recolha de medidas:

result = omlc_start();

Sendo o output de *omlc_start()* -1 caso haja algum problema com a configuração, indicando um erro, ou 0 caso não exista nenhum problema e a recolha de dados tenha sido iniciada com sucesso.

5 – Chamar *omlc_inject()* cada vez que se queira guardar uma amostra de medida:

*void omlc_inject (OmlMP *mp, OmlValueU *values);*

6 – Chamar omlc_close() para terminar o processo de colecção de medidas:

omlc_close();

7 – Compilar o programa, com link para a biblioteca *liboml2*:

$ gcc -c my-oml-client.c

$ gcc -o my-oml-client my-oml-client.o -loml2

Apêndice B – Exemplo de Resultados de *Iperf*

[4] local 192.168.0.3 port 5001 connected with 192.168.0.1 port 43652

[ID] Interval	Transfer	Bandwidth	Jitter	Lost/Total Datagrams
[4] 0.0- 5.0 sec	640 KBytes	1.05 Mbits/sec	0.020 ms	0/ 446 (0%)
[4] 5.0-10.0 sec	640 KBytes	1.05 Mbits/sec	0.010 ms	0/ 446 (0%)
[4] 10.0-15.0 sec	639 KBytes	1.05 Mbits/sec	0.005 ms	0/ 445 (0%)
[4] 15.0-20.0 sec	640 KBytes	1.05 Mbits/sec	0.010 ms	0/ 446 (0%)
[4] 20.0-25.0 sec	640 KBytes	1.05 Mbits/sec	0.003 ms	0/ 446 (0%)
[4] 25.0-30.0 sec	640 KBytes	1.05 Mbits/sec	0.005 ms	0/ 446 (0%)
[4] 30.0-35.0 sec	640 KBytes	1.05 Mbits/sec	0.074 ms	0/ 446 (0%)
[4] 35.0-40.0 sec	640 KBytes	1.05 Mbits/sec	0.003 ms	0/ 446 (0%)
[4] 40.0-45.0 sec	640 KBytes	1.05 Mbits/sec	0.080 ms	0/ 446 (0%)
[4] 45.0-50.0 sec	639 KBytes	1.05 Mbits/sec	0.031 ms	0/ 445 (0%)
[4] 50.0-55.0 sec	640 KBytes	1.05 Mbits/sec	0.014 ms	0/ 446 (0%)
[4] 55.0-60.0 sec	640 KBytes	1.05 Mbits/sec	0.011 ms	0/ 446 (0%)
[4] 0.0-60.0 sec	7.50 MBytes	1.05 Mbits/sec	0.013 ms	0/ 5351 (0%)

Apêndice C – Experiência com MGEN

0.0 ON 1 UDP SRC 5001 DST 10.110.1.2/5000 PERIODIC [10000 4096]

15.0 ON 2 UDP SRC 5002 DST 10.110.1.2/5000 PERIODIC [5000 4096]

30.0 ON 3 UDP SRC 5003 DST 10.110.1.2/5000 PERIODIC [5000 4096]

45.0 ON 4 UDP SRC 5004 DST 10.110.1.2/5000 PERIODIC [5000 4096]

60.0 OFF 1

60.0 OFF 2

60.0 OFF 3

60.0 OFF 4

Apêndice D – Experiência com D-ITG

EXECUÇÃO

1. Dar início ao ITGRecv no nó destinatário (192.168.1.3)
[root@amazing]$ ./ITGRecv -l recv_log_file
2. Dar início ao ITGSend no nó remetente (192.168.1.2)
[root@amazing]$ ./ITGSend -a 192.168.1.3 -rp 9501 -C 1000 –u 500 1000 -l send_log_file
3. Fechar o ITGRecv pressionando Ctrl-C
4. Analisar o registo no nó de destino:
[root@amazing]$ ./ITGDec recv_log_file

RESULTADOS

Flow number: 1
From 192.168.1.2:34771
To 192.168.1.3:9501
Total time = 10.001837 s
Total packets = 10000
Minimum delay = 3633.445701 s
Maximum delay = 3633.464808 s
Average delay = 3633.449749 s
Average *jitter* = 0.000706 s
Delay standard deviation = 0.001364 s
Bytes received = 7498028
Average bitrate = 5997.320692 Kbit/s
Average packet rate = 999.816334 pkt/s
Packets dropped = 0 (0 %)

********** TOTAL RESULTS **********
Number of flows = 1
Total time = 10.001837 s
Total packets = 10000
Minimum delay = 3633.445701 s
Maximum delay = 3633.464808 s
Average delay = 3633.449749 s
Average *jitter* = 0.000706 s
Delay standard deviation = 0.036939 s
Bytes received = 7498028
Average bitrate = 5997.320692 Kbit/s
Average packet rate = 999.816334 pkt/s
Packets dropped = 0 (0 %)
Error lines = 0
5. Analisar o registo no nó de origem:
[root@amazing]$ ./ITGDec send_log_file

Flow number: 1
From 10.0.0.4:34771

To 10.0.0.3:9501

Total time = 9.999002 s
Total packets = 10000
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average *jitter* = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 7498028
Average bitrate = 5999.021102 Kbit/s
Average packet rate = 1000.099810 pkt/s
Packets dropped = 0 (0 %)

********** TOTAL RESULTS **********
Number of flows = 1
Total time = 9.999002 s
Total packets = 10000
Minimum delay = 0.000000 s
Maximum delay = 0.000000 s
Average delay = 0.000000 s
Average *jitter* = 0.000000 s
Delay standard deviation = 0.000000 s
Bytes received = 7498028
Average bitrate = 5999.021102 Kbit/s
Average packet rate = 1000.099810 pkt/s
Packets dropped = 0 (0 %)
Error lines = 0

Apêndice E – Integração de Exportador na OML

Definição da Aplicação

```
1  # This is an OMF Definition for an existing application called ‘nprobe’
2  # This definition will allow OMF entities to use and instrument this application
3
4  defApplication('nprobe', 'nprobeWrapper') {|app|
5
6  app.shortDescription = "nProbe Wrapper"
7  app.path="/usr/local/bin/nprobe"
8
9  # Define the properties that can be configure for this application
10
11 app.defProperty(‘collector_host’, ‘IP adress of the collector’, ?n, {:type => :string, 12
:dynamic => false})
13 app.defProperty(‘flow_version’, ‘Netflow/IPFIX version’, ?v, {:type => :integer,
14 :dynamic => false})
15 app.defProperty(‘interface’, ‘Interface to capture data from’, ?i, {:type => string,
16 :dynamic => false})
17 app.defProperty(‘template_definition’, ‘IFIX Template’, ?T, {:type => string,
18 :dynamic => false})
19 app.defProperty(‘verbose_level’, ‘Verbose level’, ?b, {:type => integer, :dynamic 20  =>
false})
21
22 }
```

Definição do Protótipo

```
1 # Define a new prototype
2 # This is an OMF Prototype definition is an instance/specialization of the above
3 # 'nprobeWrapper' application
4
5 defPrototype("ipfix_nprobe") { |proto|
```

```
6
7  proto.name = "IPFIX_nprobe"
8  proto.description = "An IPFIX probe - nProbe"
9
10  # Define the properties (and their default values) that can be configured for this
11  # prototype
12
13  proto.defProperty('collector', 'IP address of the collector', '10.110.1.150:4739')
14  proto.defProperty('flow', 'Netflow/IPFIX version', 10)
20  proto.defProperty('int, 'Interface to capture data from')
21  proto.defProperty('template', 'IPFIX template')
22  proto.defProperty('verbose', 'Verbose level', 3)
23
24  # Here we bind this prototype with the "nprobeWrapper" application definition
25  # And we also bind the properties that we decided to use and gave default values to
26
27   proto.addApplication("nprobe", "nprobeWrapper") { |ipfix|
28    ipfix.bindProperty('collector_host', 'collector')
29    ipfix.bindProperty('flow_version', 'flow')
20    ipfix.bindProperty('interface, 'int')
31    ipfix.bindProperty('template_definition', 'template')
32    ipfix.bindProperty('verbose_level', 'verbose')
33   }
34 }
```

Definição da Experiência

```
1  #
2  # A)  Define the 'source' group, which has the unique node [1,1]
3  #       Nodes in this group will execute the application 'test:proto:iperfudpsender'
4
5  defGroup('source', [1,1]) {|node|
6
7  node.prototype("test:proto:iperfudpsender", {
```

```
8    'port' => '5000',
9    'client' => '192.168.0.3'
10 })
12
13 # B)  Define the 'sink' group, which has the unique node [1,3]
14 #      Nodes in this group will execute the application 'test:proto:iperfudpreceiver'
15 #      and 'ipfix_nprobe'
16
17 defGroup('sink', [1,3]) {|node|
18
19 node.prototype("test:proto:iperfudpreceiver",{
20   'port' => '5000'
21 })
22
23 node.prototype("ipfix_nprobe", {
24    'interf' => 'ath1',
25    'template' => '-T "%IPV4_SRC_ADDR %IPV4_DST_ADDR
26                    %IN_BYTES %FIRST_SWITCHED %LAST_SWITCHED
27                    %PROTOCOL %RTP_FIRST_SSRC % RTP_FIRST_TS
28                    %RTP_LAST_SSRC %RTP_LAST_TS %RTP_IN_JITTER
29                    %RTP_OUT_JITTER %RTP_IN_PKT_LOST
30                    %RTP_OUT_PKT_LOST %RTP_OUT_PAYLOAD_TYPE
31                    %RTP_IN_MAX_DELTA %RTP_OUT_MAX_DELTA
32 })
33
34 # C)  Configure the wireless interfaces of All the Nodes involved in this experiment
35
36 allGroups.net.w0 { |w|
37   w.type = 'g'
38   w.channel = "6"
39   w.essid = "amazing"
40   w.ip = "%192.168.0.%y"
41   w.mode = "adhoc"
42 }
```

```
43
44 # D)  When all the nodes are turned on and the all the applications
45 #       are installed and ready, we can start to perform the experiment
46
47 whenAllInstalled() {|node|
48
49 wait 30
50
51 allGroups.startApplications
52 wait 60
53
54 Experiment.done
55 }
```

Apêndice F – Métricas da *nProbe*

A informação contida neste apêndice é retirada de [37].

ID Flow Label Description

---

[1] %IN_BYTES Incoming flow bytes (src->dst)

[1] %SYSTEM_ID

[2] %IN_PKTS Incoming flow packets (src->dst)

[2] %INTERFACE_ID

[3] %FLOWS Number of flows

[3] %LINE_CARD

[4] %PROTOCOL IP protocol byte

[164] %PROTOCOL_MAP IP protocol name

[4] %NETFLOW_CACHE

[5] %SRC_TOS Type of service byte

[5] %TEMPLATE_ID

[6] %TCP_FLAGS Cumulative of all flow TCP flags

[7] %L4_SRC_PORT IPv4 source port

[167] %L4_SRC_PORT_MAP IPv4 source port symbolic name

[8] %IPV4_SRC_ADDR IPv4 source address

[9] %IPV4_SRC_MASK IPv4 source subnet mask (/<bits>)

[10] %INPUT_SNMP Input interface SNMP idx

[11] %L4_DST_PORT IPv4 destination port

[171] %L4_DST_PORT_MAP IPv4 destination port symbolic name

[12] %IPV4_DST_ADDR IPv4 destination address

[13] %IPV4_DST_MASK IPv4 dest subnet mask (/<bits>)

[14] %OUTPUT_SNMP Output interface SNMP idx

[15] %IPV4_NEXT_HOP IPv4 next hop address

[16] %SRC_AS Source BGP AS

[17] %DST_AS Destination BGP AS

[21] %LAST_SWITCHED SysUptime (msec) of the last flow pkt

[22] %FIRST_SWITCHED SysUptime (msec) of the first flow pkt

[23] %OUT_BYTES Outgoing flow bytes (dst->src)

[24] %OUT_PKTS Outgoing flow packets (dst->src)

[27] %IPV6_SRC_ADDR IPv6 source address

[28] %IPV6_DST_ADDR IPv6 destination address

[29] %IPV6_SRC_MASK IPv6 source mask

[30] %IPV6_DST_MASK IPv6 destination mask

[32] %ICMP_TYPE ICMP Type * 256 + ICMP code

[34] %SAMPLING_INTERVAL Sampling rate

[35] %SAMPLING_ALGORITHM Sampling type (deterministic/random)

[36] %FLOW_ACTIVE_TIMEOUT Activity timeout of flow cache entries

[37] %FLOW_INACTIVE_TIMEOUT Inactivity timeout of flow cache entries

[38] %ENGINE_TYPE Flow switching engine

[39] %ENGINE_ID Id of the flow switching engine

[40] %TOTAL_BYTES_EXP Total bytes exported

[41] %TOTAL_PKTS_EXP Total flow packets exported

[42] %TOTAL_FLOWS_EXP Total number of exported flows

[56] %IN_SRC_MAC Source MAC Address

[57] %OUT_DST_MAC Destination MAC Address

[58] %SRC_VLAN Source VLAN

[59] %DST_VLAN Destination VLAN

[60] %IP_PROTOCOL_VERSION [4=IPv4][6=IPv6]

[61] %DIRECTION [0=ingress][1=egress] flow

[62] %IPV6_NEXT_HOP IPv6 next hop address

[70] %MPLS_LABEL_1 MPLS label at position 1

[71] %MPLS_LABEL_2 MPLS label at position 2

[72] %MPLS_LABEL_3 MPLS label at position 3

[73] %MPLS_LABEL_4 MPLS label at position 4

[74] %MPLS_LABEL_5 MPLS label at position 5

[75] %MPLS_LABEL_6 MPLS label at position 6

[76] %MPLS_LABEL_7 MPLS label at position 7

[77] %MPLS_LABEL_8 MPLS label at position 8

[78] %MPLS_LABEL_9 MPLS label at position 9

[79] %MPLS_LABEL_10 MPLS label at position 10

[148] %FLOW_ID Serial Flow Identifier

[NFv9 57552][IPFIX 35632.80] %FRAGMENTS Number of fragmented flow packets

[NFv9 57554][IPFIX 35632.82] %CLIENT_NW_DELAY_SEC Network latency client <-> nprobe (sec)

[NFv9 57555][IPFIX 35632.83] %CLIENT_NW_DELAY_USEC Network latency client <-> nprobe (usec)

[NFv9 57556][IPFIX 35632.84] %SERVER_NW_DELAY_SEC Network latency nprobe <-> server (sec)

[NFv9 57557][IPFIX 35632.85] %SERVER_NW_DELAY_USEC Network latency nprobe <-> server (usec)

[NFv9 57558][IPFIX 35632.86] %APPL_LATENCY_SEC Application latency (sec)

[NFv9 57559][IPFIX 35632.87] %APPL_LATENCY_USEC Application latency (usec)

[NFv9 57570][IPFIX 35632.98] %ICMP_FLAGS Cumulative of all flow ICMP types

[NFv9 57573][IPFIX 35632.101] %SRC_IP_COUNTRY Country where the src IP is located

[NFv9 57574][IPFIX 35632.102] %SRC_IP_CITY City where the src IP is located

[NFv9 57575][IPFIX 35632.103] %DST_IP_COUNTRY Country where the dst IP is located

[NFv9 57576][IPFIX 35632.104] %DST_IP_CITY City where the dst IP is located

[NFv9 57577][IPFIX 35632.105] %FLOW_PROTO_PORT L7 port that identifies the flow protocol or 0 if unknown

[NFv9 57578][IPFIX 35632.106] %TUNNEL_ID Tunnel identifier (e.g. GTP tunnel Id) or 0 if unknown

[NFv9 57579][IPFIX 35632.107] %LONGEST_FLOW_PKT Longest packet (bytes) of the flow

[NFv9 57580][IPFIX 35632.108] %SHORTEST_FLOW_PKT Shortest packet (bytes) of the flow

[NFv9 57581][IPFIX 35632.109] %RETRANSMITTED_IN_PKTS Number of retransmitted TCP flow packets (src->dst)

[NFv9 57582][IPFIX 35632.110] %RETRANSMITTED_OUT_PKTS Number of retransmitted TCP flow packets (dst->src)

[NFv9 57583][IPFIX 35632.111] %OOORDER_IN_PKTS Number of out of order TCP flow packets (dst->src)

[NFv9 57584][IPFIX 35632.112] %OOORDER_OUT_PKTS Number of out of order TCP flow packets (dst->src)

[NFv9 57585][IPFIX 35632.113] %UNTUNNELED_PROTOCOL Untunneled IP protocol byte

[NFv9 57586][IPFIX 35632.114] %UNTUNNELED_IPV4_SRC_ADDR Untunneled IPv4 source address

[NFv9 57587][IPFIX 35632.115] %UNTUNNELED_L4_SRC_PORT Untunneled IPv4 source port

[NFv9 57588][IPFIX 35632.116] %UNTUNNELED_IPV4_DST_ADDR Untunneled IPv4 destination address

[NFv9 57589][IPFIX 35632.117] %UNTUNNELED_L4_DST_PORT Untunneled IPv4 destination port

Plugin BGP Update Listener templates:

[NFv9 57762][IPFIX 35632.290] %SRC_AS_PATH_1 Src AS path position 1

[NFv9 57763][IPFIX 35632.291] %SRC_AS_PATH_2 Src AS path position 2

[NFv9 57764][IPFIX 35632.292] %SRC_AS_PATH_3 Src AS path position 3

[NFv9 57765][IPFIX 35632.293] %SRC_AS_PATH_4 Src AS path position 4

[NFv9 57766][IPFIX 35632.294] %SRC_AS_PATH_5 Src AS path position 5

[NFv9 57767][IPFIX 35632.295] %SRC_AS_PATH_6 Src AS path position 6

[NFv9 57768][IPFIX 35632.296] %SRC_AS_PATH_7 Src AS path position 7

[NFv9 57769][IPFIX 35632.297] %SRC_AS_PATH_8 Src AS path position 8

[NFv9 57770][IPFIX 35632.298] %SRC_AS_PATH_9 Src AS path position 9

[NFv9 57771][IPFIX 35632.299] %SRC_AS_PATH_10 Src AS path position 10

[NFv9 57772][IPFIX 35632.300] %DST_AS_PATH_1 Dest AS path position 1

[NFv9 57773][IPFIX 35632.301] %DST_AS_PATH_2 Dest AS path position 2

[NFv9 57774][IPFIX 35632.302] %DST_AS_PATH_3 Dest AS path position 3

[NFv9 57775][IPFIX 35632.303] %DST_AS_PATH_4 Dest AS path position 4

[NFv9 57776][IPFIX 35632.304] %DST_AS_PATH_5 Dest AS path position 5

[NFv9 57777][IPFIX 35632.305] %DST_AS_PATH_6 Dest AS path position 6

[NFv9 57778][IPFIX 35632.306] %DST_AS_PATH_7 Dest AS path position 7

[NFv9 57779][IPFIX 35632.307] %DST_AS_PATH_8 Dest AS path position 8

[NFv9 57780][IPFIX 35632.308] %DST_AS_PATH_9 Dest AS path position 9

[NFv9 57781][IPFIX 35632.309] %DST_AS_PATH_10 Dest AS path position 10

Plugin DNS Protocol Dissector templates:

[NFv9 57677][IPFIX 35632.205] %DNS_QUERY DNS QUERY

[NFv9 57678][IPFIX 35632.206] %DNS_QUERY_ID DNS query transaction Id

[NFv9 57679][IPFIX 35632.207] %DNS_QUERY_TYPE DNS query type (e.g. 1=A, 2=NS..)

[NFv9 57680][IPFIX 35632.208] %DNS_RET_CODE DNS return code (e.g. 0=no error)

[NFv9 57681][IPFIX 35632.209] %DNS_NUM_ANSWER DNS # of returned answers

Plugin dump templates:

[NFv9 57592][IPFIX 35632.120] %DUMP_PATH Path where dumps will be saved

Plugin HTTP Protocol Dissector templates:

[NFv9 57652][IPFIX 35632.180] %HTTP_URL HTTP URL

[NFv9 57653][IPFIX 35632.181] %HTTP_RET_CODE HTTP return code (e.g. 200, 304...)

[NFv9 57654][IPFIX 35632.182] %HTTP_REFERER HTTP Referer

[NFv9 57655][IPFIX 35632.183] %HTTP_UA HTTP User Agent

[NFv9 57656][IPFIX 35632.184] %HTTP_MIME HTTP Mime Type

Plugin L7 Protocol Recognition templates:

[NFv9 57637][IPFIX 35632.165] %L7_PROTO Symbolic layer 7 protocol description

Plugin MySQL Plugin templates:

[NFv9 57667][IPFIX 35632.195] %MYSQL_SERVER_VERSION MySQL server version

[NFv9 57668][IPFIX 35632.196] %MYSQL_USERNAME MySQL username

[NFv9 57669][IPFIX 35632.197] %MYSQL_DB MySQL database in use

[NFv9 57670][IPFIX 35632.198] %MYSQL_QUERY MySQL Query

[NFv9 57671][IPFIX 35632.199] %MYSQL_RESPONSE MySQL server response

Plugin RTP templates:

[NFv9 57622][IPFIX 35632.150] %RTP_FIRST_SSRC First flow RTP Sync Source ID

[NFv9 57623][IPFIX 35632.151] %RTP_FIRST_TS First flow RTP timestamp

[NFv9 57624][IPFIX 35632.152] %RTP_LAST_SSRC Last flow RTP Sync Source ID

[NFv9 57625][IPFIX 35632.153] %RTP_LAST_TS Last flow RTP timestamp

[NFv9 57626][IPFIX 35632.154] %RTP_IN_JITTER RTP Jitter (ms * 1000)

[NFv9 57627][IPFIX 35632.155] %RTP_OUT_JITTER RTP Jitter (ms * 1000)

[NFv9 57628][IPFIX 35632.156] %RTP_IN_PKT_LOST Packet lost in stream

[NFv9 57629][IPFIX 35632.157] %RTP_OUT_PKT_LOST Packet lost in stream

[NFv9 57630][IPFIX 35632.158] %RTP_OUT_PAYLOAD_TYPE RTP payload type

[NFv9 57631][IPFIX 35632.159] %RTP_IN_MAX_DELTA Max delta (ms*100) between consecutive pkts

[NFv9 57632][IPFIX 35632.160] %RTP_OUT_MAX_DELTA Max delta (ms*100) between consecutive pkts

Plugin SIP templates:

[NFv9 57602][IPFIX 35632.130] %SIP_CALL_ID SIP call-id

[NFv9 57603][IPFIX 35632.131] %SIP_CALLING_PARTY SIP Call initiator

[NFv9 57604][IPFIX 35632.132] %SIP_CALLED_PARTY SIP Called party

[NFv9 57605][IPFIX 35632.133] %SIP_RTP_CODECS SIP RTP codecs

[NFv9 57606][IPFIX 35632.134] %SIP_INVITE_TIME SIP SysUptime (msec) of INVITE

[NFv9 57607][IPFIX 35632.135] %SIP_TRYING_TIME SIP SysUptime (msec) of Trying

[NFv9 57608][IPFIX 35632.136] %SIP_RINGING_TIME SIP SysUptime (msec) of RINGING

[NFv9 57609][IPFIX 35632.137] %SIP_OK_TIME SIP SysUptime (msec) of OK

[NFv9 57610][IPFIX 35632.138] %SIP_BYE_TIME SIP SysUptime (msec) of BYE

[NFv9 57611][IPFIX 35632.139] %SIP_RTP_SRC_IP SIP RTP stream source IP

[NFv9 57612][IPFIX 35632.140] %SIP_RTP_SRC_PORT SIP RTP stream source port

[NFv9 57613][IPFIX 35632.141] %SIP_RTP_DST_IP SIP RTP stream dest IP

[NFv9 57614][IPFIX 35632.142] %SIP_RTP_DST_PORT SIP RTP stream dest port

Plugin SMTP Protocol Dissector templates:

[NFv9 57657][IPFIX 35632.185] %SMTP_MAIL_FROM Mail sender

[NFv9 57658][IPFIX 35632.186] %SMTP_RCPT_TO Mail recipient